UNIVERSIDADE FEDERAL DO PARANÁ
ADEILDO FERNANDES DOS SANTOS

DESENVOLVIMENTO DE UMA ARQUITETURA DE ARMAZENAMENTO HÍBRIDA LINHA-COLUNA PARA O GERENCIAMENTO DE SENSORES DE INSTRUMENTAÇÃO DE BARRAGENS

CURITIBA
2012

ADEILDO FERNANDES DOS SANTOS

DESENVOLVIMENTO DE UMA ARQUITETURA DE ARMAZENAMENTO HÍBRIDA LINHA-COLUNA PARA O GERENCIAMENTO DOS SENSORES DE INSTRUMENTAÇÃO DE BARRAGENS

Monografia apresentada como requisito parcial para a obtenção da titulação de especialista, pelo Curso de Pós-Graduação Lato Sensu em Engenharia de Software, da Universidade Federal do Paraná.

Orientador: Professor Jaime Wojciechowski

CURITIBA

2012

TERMO DE APROVAÇÃO

ADEILDO FERNANDES DOS SANTOS

DESENVOLVIMENTO DE UMA ARQUITETURA DE ARMAZENAMENTO HÍBRIDA LINHA-COLUNA PARA O GERENCIAMENTO DOS SENSORES DE INSTRUMENTAÇÃO DE BARRAGENS

Monografia apresentada como requisito parcial para a obtenção da titulação de especialista, pelo Curso de Pós-Graduação Lato Sensu em Engenharia de Software, da Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientador: Professor Jaime Wojciechowski

Curitiba, 18 de dezembro de 2012

RESUMO

Sistemas computacionais tem evoluído rapidamente nos últimos anos, tanto em relação ao hardware quanto aos requisitos de software. Essa evolução tem permitido a geração/armazenamento de volumes de dados cada vez maiores. Para a utilização completa e adequada a dessa grande massa de dados, sistemas computacionais tem feito uso de cargas de trabalho heterogêneas, sendo elas transacionais e analíticas. Essas cargas de trabalho de natureza distinta exigem que as arquiteturas de armazenamento de dados sejam bem projetadas, a fim de que haja uma boa performance para a utilização do sistemas que fazem seu uso. Buscando contribuir para o bom funcionamento desses sistemas, neste trabalho é apresentado uma proposta de arquitetura híbrida linha-coluna para o armazenamento de dados. Esta arquitetura é baseada no modelo relacional e no modelo orientado a coluna, e pretende beneficiar as variadas cargas de trabalho, independente de sua natureza. Para validar a arquitetura proposta foi desenvolvido um sistema de monitoramento de sensores de instrumentação de uma barragem hidrelétrica, cujas características evidenciam e justificam o uso de uma abordagem diferenciada em relação ao armazenamento e recuperação dos seus dados.

Palavras-chaves: Column-Store, Modelo Relacional, Monitoramento de sensores

ABSTRACT

Computer systems have evolved rapidly in recent years, both in terms of hardware and software requirements. This evolution has allowed the generation and storage data volumes increasing. For the full and proper use of this large amount of data, computer systems have made use of heterogeneous workloads, whether they are transacional and analytical. These workloads of different nature require data storage architectures are well designed, so that there is good performance for using the systems that make its use. In order to contribute to the proper functioning of these systems, this paper presents a proposal for a hybrid architecture for row-column data storage. This architecture is based on the relational model and the column-oriented model, and want to benefit from the varied workloads, regardless of their nature. To validate the architecture proposed we developed a monitoring system sensors instrumentation of a hydroelectric dam, whose features show and justify the use of a differentiated approach to the storage and retrieval of their data.

Key-words: Column-Store, relational model, Monitoring sensors

SUMÁRIO

1 INTRODUÇÃO

1.1 INTRODUCÃO

Ao longo das últimas décadas diversos aspectos fundamentais relacionados aos Sistemas Gerenciadores de Banco de Dados (SGBDs), tais como mudanças arquiteturais, necessidade de desempenho e tipos de carga de trabalho, têm mudado drasticamente. O aumento da velocidade das unidades de processamento, acompanhado pelo aumento da capacidade das memórias principais e dos discos, e a inclusão de vários níveis de memória *cache* são fatores que têm melhorado o desempenho de sistemas de banco de dados.

Embora a capacidade dos discos tenha aumentado (acompanhado pelo decréscimo de seus preços), as velocidades de acesso e transferências de dados não tiveram uma melhora significativa. Isso faz com que as taxas de Entrada e Saída (E/S) constituam o principal gargalo em um SGBD (MANEGOLD, 2000) tornando este um dos principais problemas de desempenho a serem tratados em um projeto de SGBD.

Além de se adaptar as evoluções de *Hardware*, os SGBDs devem se adequar as novas aplicações que fazem seu uso, as quais tem emergido com requisitos diversificados em relação a aqueles encontrados em sistemas transacionais tradicionais. Para citar algumas dessas aplicações temos os sistemas de *Data Warehouse*, *Data Mining* e *Business Inteligence*.

Essas novas aplicações se caracterizam pelo uso extensivo de cargas de trabalho orientadas a leitura e que são constituídas de consultas complexas que manipulam grandes massas de dados e fazem uso de diversas funções de agregação, sendo caracterizadas como analíticas.

Sendo assim, em um bom projeto arquitetural de armazenamento de dados deve haver a preocupação de se utilizar de maneira eficiente os recursos de *hardware* disponíveis, assim como possibilitar a operação eficiente tanto de cargas de trabalho transacionais quanto analíticas.

Atualmente a maior parte dos SGBDs emprega o sistema de armazenamento

NSM, também conhecido como orientado à linha, para persistir os registros de uma relação em disco. Nesse esquema de armazenamento físico todos os atributos de um registro são colocados de maneira contínua em disco, ou linha a linha (RAMAKRISHNAN, 2000).

Essa estratégia de alocação física de dados oferece boa performance quando a consulta a ser submetida necessitar realizar acesso total aos dados de um registro, ou quando acessos aleatórios devem ser realizados. Essas características são frequentes em consultas do tipo transacional, fazendo com que para esse tipo de carga de trabalho a orientação à linha seja a estratégia mais adequada a ser implementada.

No entanto nem todas as consultas fazem uso integral dos atributos de um registro, necessitando apenas de uma fração de seus dados. Para esses casos onde acessos parciais são necessários o modelo orientado à linha torna-se ineficaz pois tem que carregar do disco para a memória, dados que não são necessário, aumentando assim a quantidade de E/S a ser realizada (COPELAND, 1987).

Para minimizar essa E/S desnecessária foi proposto o modelo DSM (COPELAND, 1985), frequentemente chamado de orientado à coluna, o qual armazena cada um dos atributos de uma relação de maneira contínua em disco, ou coluna a coluna. Essa característica permite que somente os dados requisitados em uma consulta sejam carregados para memória em consultas que façam acesso parcial aos registros. Cargas de trabalho analíticas e consultas que envolvam agregações geralmente possuem essas característica, fazendo com que a orientação à coluna seja adequada para essas cargas de trabalho. Em sistemas orientados a coluna a performance de E/S pode ser melhorada ainda mais em função destes poderem trabalhar facilmente com dados comprimidos.

Assim como o modelo orientado à linha, a orientação à coluna possui algumas desvantagens. Em sistemas orientados à coluna para se realizar um acesso total a um determinado registro é necessário buscar cada um dos seus atributos (que se encontram em lugares distintos do disco) a fim de poder reconstruí-lo.

A reconstrução de um registro implica na realização de diversas junções *(joins),* ocasionando um alto custo de processamento pelo SGBD e uma

performance pobre para o sistema (ABADI, 2006). Operações com inserções, exclusões e atualizações (quando estas fizerem acesso a grande parte dos atributos) são onerosas na orientação à coluna pelo mesmo fato. Uma relação que possui *n* atributos implica em *n* acessos a disco para localizar o dado alvo da operação, fazendo com que sistemas orientados à linha sejam ineficientes para cargas de trabalho transacionais.

Diante do contexto apresentado pode-se observar que tanto a orientação à linha quanto a orientação à coluna não são eficientes para todos os tipos de consultas. Atualmente a maior parte das aplicações utilizam somente um dos esquemas de armazenamento físico de dados, fazendo com que cada um desses sistemas seja utilizado de forma eficiente para determinadas tarefas, sendo otimizados para algumas cargas de trabalho em detrimento de outras e nunca sendo ideal para todas.

Considerando os fatos apresentados a principal abordagem a ser considerada para propor uma solução que mantenha uma boa performance, independente da carga de trabalho a ser processada, seria implementar uma arquitetura de armazenamento de dados híbrida que utiliza ambas as estratégias de armazenamento físico, de maneira transparente ao usuário final.

Esta estratégia consistirá em colocar simultaneamente parte das relações a serem persistidas no banco com uma cópia armazenada em linha e outra em coluna. Esta arquitetura proverá a possibilidade de obter os dados do modelo de armazenamento que seja mais proveitoso para as características da consulta a ser executada.

Um detalhe importante a ser considerado é que a performance dos comandos de Inserção, Exclusão e Atualização quando submetidas em um banco que utiliza armazenamento em coluna, em geral, é inferior quando comparada ao armazenamento em linha, tornando a preferência da execução deste tipo de consulta sempre na relação orientada à linha. Para garantir a integridade dos dados, visto essa diferença de performance, deverá ser incluído na arquitetura híbrida de armazenamento um mecanismo de sincronização de dados.

1.2 OBJETIVOS DO PROJETO

O principal objetivo deste trabalho é utilizar o armazenamento em linha juntamente com o armazenamento em coluna de dados em disco para aumentar a performance das consultas a serem submetidas ao banco de dados, independente de sua natureza e características. Sendo assim, para tentar atingir esse objetivo será desenvolvida uma arquitetura de armazenamento híbrida linha coluna.

Para testar a eficiência da arquitetura proposta será utilizado como estudo de caso o desenvolvimento de um sistema para o gerenciamento das leituras realizadas pelos sensores de instrumentação de barragens hidrelétricas. As tecnologias empregadas na implementação do sistema consistirão na linguagem de programação Java, o *framework* para desenvolvimento de aplicações *web* JSF juntamente com o banco de dados MySQL.

Deste modo, o restante deste trabalho está organizado da seguinte maneira: O Capítulo 2 irá apresentar os conceitos fundamentais para o entendimento da solução proposta, discutindo sobre os modelos de armazenamento físico e suas principais características. O Capítulo 3 apresentará as ideias gerais do estudo de caso que será utilizado para validar a abordagem proposta e alguns pontos importantes a serem considerados na implementação. O capítulo 4 apresentará respectivamente o software desenvolvido e os testes de performance realizados sobre o mesmo. E por fim, no capítulo 5 será realizada uma discussão sobre os resultados obtidos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 REVISÃO BIBLIOGRÁFICA

Este Capítulo tem como principal finalidade fornecer um referencial teórico que possibilite uma melhor compreensão da proposta e dos objetivos que pretendem ser alcançados neste trabalho.

Primeiramente serão descritos os principais modelos de armazenamento físico de dados e posteriormente será aprofundada a discussão sobre um desses modelos, o armazenamento em coluna. Em seguida serão apresentados e discutidos os principais tipos de carga de trabalho que são executadas em banco de dados.

2.2 MODELOS DE ARMAZENAMENTO DE DADOS EM DISCO

Nas últimas décadas, em função de diversas mudanças arquiteturais de seus componentes, computadores têm melhorado seu desempenho. As CPUs estão ficando cada vez mais rápidas, memórias e discos maiores em capacidade de armazenamento e com custos menores. Embora estes dispositivos estejam ficando cada vez mais sofisticados um problema que afeta a performance global de um sistema de banco de dados ainda persiste, o gargalo de E/S entre memória principal e disco.

Na tentativa de diminuir esse gargalo foram propostas mudanças em relação ao armazenamento físico de dados em disco. Essas mudanças ocasionaram melhoras de desempenho em função da diminuição na quantidade de acessos a disco necessárias, melhorando substancialmente o tempo de resposta das consultas. A partir dessas mudanças surgiu o modelo DSM. Detalhes sobre este modelo e também do modelo tradicional de armazenamento (NSM) serão abordados nas próximas seções.

2.2.1 NSM

Atualmente a maior parte dos Sistemas Gerenciadores de Banco de Dados utilizam o *N-ary Storage Model* (NSM) (RAMAKRISHNAN, 2000) como modelo de armazenamento físico de dados em disco. Neste modelo de armazenamento todos os atributos de um registro são colocados de maneira contígua em disco. A Figura 2.1 apresenta a maneira na qual o NSM armazena os registros de uma relação.

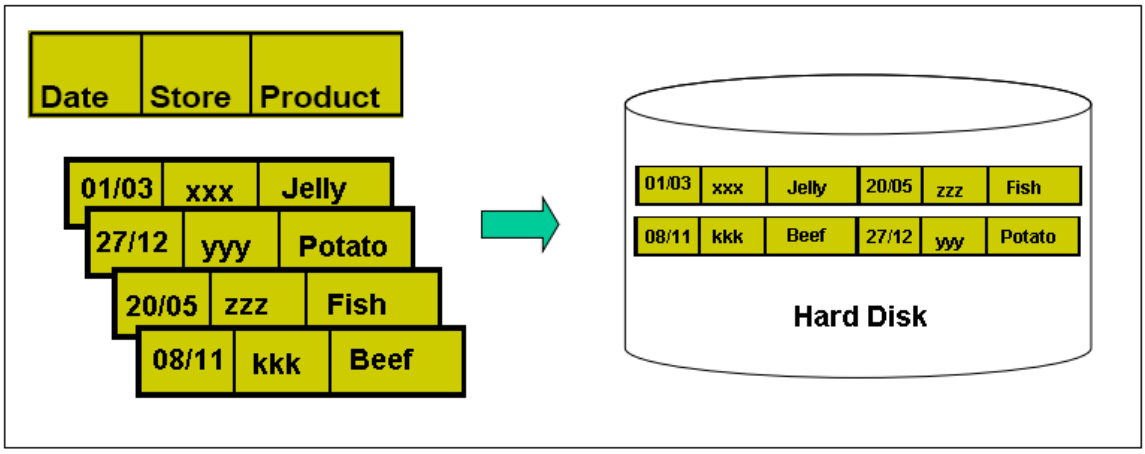


Figura 2.1: NSM - *N-ary Storage Model*
FONTE: O autor (2012).

Em função dos dados de uma relação estarem armazenados sequencialmente em disco possibilita que consultas que façam acesso total a um registro tenham um bom desempenho. Por outro lado quando é necessário realizar acesso parcial a um registro, ou seja, quando somente alguns atributos da relação são requeridos, o mesmo tem que ser lido e transferido de maneira integral do disco para a memória principal, mesmo que a maioria dos dados carregados não sejam úteis para o processamento da consulta.

Essa característica aumenta desnecessariamente a quantidade total de acessos a disco, além de desperdiçar parte da memória principal, fazendo com que a performance de algumas consultas sejam degradadas, tornando-as ineficientes.

2.2.2 DSM

O *Decomposition Storage Model* (DSM) (COPELAND, 1985) é um modelo de armazenamento físico alternativo ao NSM. Esta nova estratégia ocasiona o particionamento vertical dos atributos de uma relação, criando sub relações binárias, onde cada uma delas possui uma chave e um valor e corresponde a cada um dos atributos da relação original. Neste estilo de armazenamento todos os valores de um atributo são armazenados consecutivamente em disco. Essa estratégia pode ser visualizada por meio da Figura 2.2.

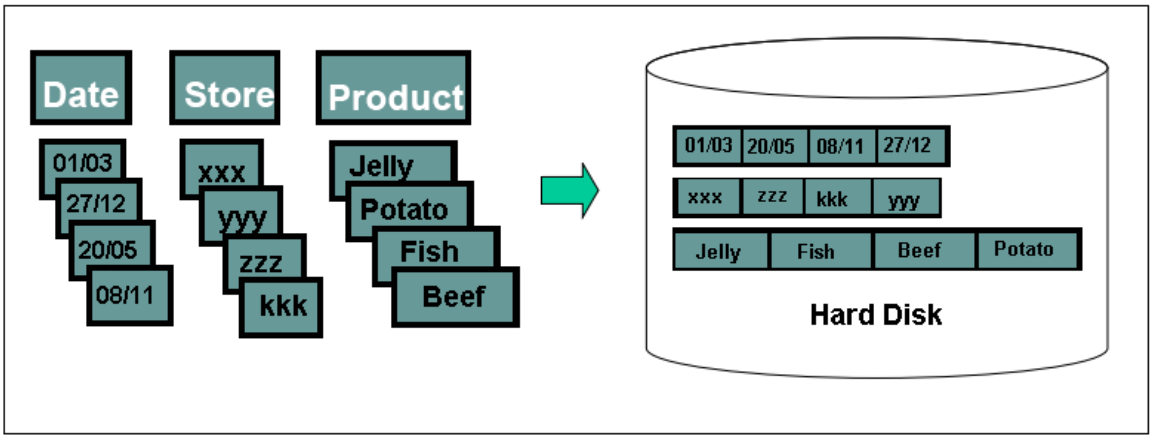


Figura 2.2: DSM - *Decomposition Storage Model*
FONTE: O autor (2012).

Para consultas que fazem acesso parcial a um registro, ou seja, quando somente uma fração dos atributos de um registro é necessário, o modelo DSM tem boa performance. Devido a sua característica de armazenar os atributos de uma relação de maneira independente, o modelo DSM permite recuperar somente as informação úteis requisitadas pela consulta, usando de maneira eficiente a memória e diminuindo a quantidade total de acessos a disco. Em função dos dados de um atributo se encontrarem armazenados juntos em disco acessos sequenciais a esses dados apresentam bom desempenho, permitindo que consultas que façam grandes acessos sequenciais, como as consultas de agregação (mim, max, sum) tenham boa performance.

O modelo DSM apresenta problemas em relação ao tempo de resposta quando as consultas a serem realizadas necessitarem da maioria dos atributos

(acesso total) de um registo. A degradação do desempenho das consultas que possuem essa natureza se dá em função do tempo necessário para realizar a reconstrução da tupla original. Acessar relações com *n* atributos gravados em DSM significa fazer *n* acessos a disco para realizar os processamentos necessários pela consulta, em função dessas características, inserções, exclusões e atualizações nesse modelo são problemáticas.

Por meio da Figura 2.3 pode-se tem uma melhor noção de como os dados em ambos os modelos de armazenamento se encontram topologicamente em disco.

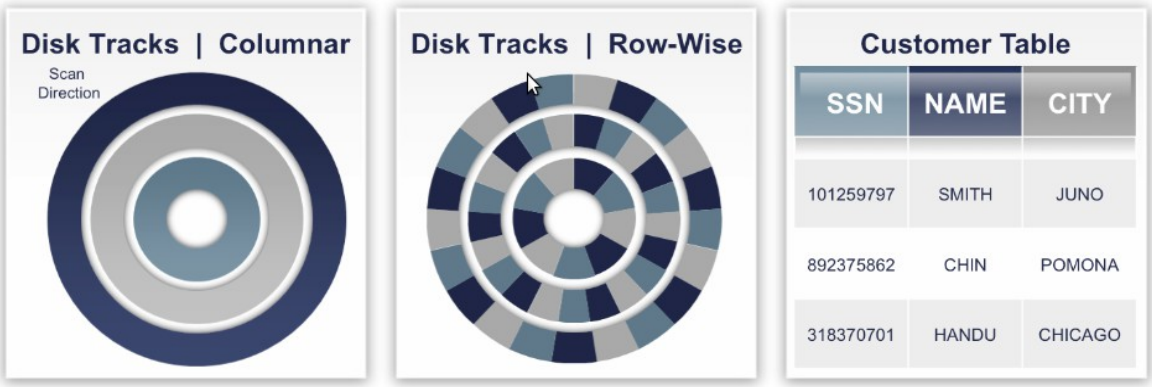


Figura 2.3: Topologia em disco modelos DSM e NSM.
FONTE: ABADI (2006).

2.3 ARMAZENAMENTO EM COLUNA

Recentemente tem existido grande interesse em armazenar relações verticalmente em disco, ou coluna a coluna, ao invés horizontalmente, ou linha a linha. A principal vantagem do armazenamento em coluna das relações é a diminuição das taxas de E/S entre disco e memória, uma vez que, grande parte das consultas submetidas ao banco de dados fazem referência a apenas uma pequena parte dos atributos de uma relação.

O DSM foi um dos primeiros modelos de armazenamento em coluna. Após o lançamento da proposta original no artigo *A Decomposition Storage Model*, diversos outros sistemas foram propostos (BONCZ, 1999), (BONCZ, 2005), (MACNICOL, 2004), (STONEBRAKER, 2005). Cada um desses sistemas possui características peculiares, mas a sua essência, o armazenamento em coluna, permanecem iguais. A cada novo sistema, características e técnicas foram incorporadas provendo cada

vez mais melhoras no modelo original.

Sendo assim, próximas seções irão tratar as diferentes maneiras de se implementar um banco de dados orientado à coluna, mostrando também alguns princípios importantes para o *design* desses sistemas. Por fim, será apresentado uma cronologia de fatos englobando desde a concepção do modelo DSM até os dias atuais, destacando alguns dos principais banco de dados orientados à coluna que foram desenvolvidos ao longo do tempo.

2.3.1 ABORDAGENS PARA O ARMAZENAMENTO DE DADOS ORIENTADO A COLUNA

Atualmente existem três abordagens descritas na literatura para implementar o modelo de armazenamento em coluna dentro de um banco de dados (ABADI, 2008). Entre estas abordagens não existem diferenças no nível lógico. A diferença entre cada uma delas se baseia apenas na maneira como os dados são alocados em disco e posteriormente são processados pelo motor de execução de consultas.

A seguir serão descritas cada uma dessas abordagens, sendo que as mesmas serão expostas segundo a sua complexidade de implementação (ABADI, 2008):

- 1° Abordagem: É a mais simples entre as três. Ela basicamente implementa o armazenamento em coluna sobre um banco de dados comum orientado à linha. Essa estratégia consiste em particionar verticalmente a relação original gerando sub relações que possuem apenas dois atributos (chave, valor). Nesta abordagem uma relação com *n* atributos resulta em *n* sub relações. Ao ser realizada uma consulta apenas as sub relações que possuem os dados a serem processados pela consulta serão acessados. A simplicidade desse processo reside no fato de que nenhuma modificação no código fonte do banco de dados é necessária. Devido a não necessidade de manipular o código fonte da aplicação esta abordagem traz a vantagem de poder ser implementada em grande parte dos SGBDs relacionais.

- 2° Abordagem: A segunda implementação consiste em modificar o gerenciador de armazenamento de dados para que ele armazene as relações coluna a coluna ao invés de linha a linha. Esta abordagem se diferencia da anterior devido a necessidade de realizar alterações no código fonte a fim de adaptar o banco de dados para que o mesmo armazene colunas ao invés de linhas. Consultas realizadas sobre banco de dados que utilizam essa abordagem devem reconstruir os registros antes que os mesmos sejam processados pelo motor de execução das consultas, pois esses são incapazes de operar sobre atributos que se encontram organizados separadamente.

- 3° Abordagem: Dentre as três abordagens esta é a mais complexa e eficiente. Aqui é necessário modificar, tanto o gerenciador de armazenamento físico de dados quanto o motor de execução das consultas. Esta implementação, além de armazenar os dados em coluna permite também processá-los da mesma maneira. Operando desse modo a necessidade de reconstrução dos registros ocorre somente ao final do plano de execução da consulta. Isso evita que resultados intermediários que não serão devolvidos ao executor da consulta sejam reconstruídos, aumentando assim a performance da consulta.

O armazenamento e processamento de dados em colunas trás diversas vantagens que provêm aumento na performance de diversas consultas (ABADI, 2008), dentre essas vantagens podemos citar, a diminuição de registros a serem gerados em operações de seleção e agregação, fazendo com que o custo final da reconstrução dos registros sejam menores.

2.2.3 CARACTERÍSTICAS DOS BANCOS DE DADOS ORIENTADOS A COLUNAS

Algumas características são especialmente importantes quando trata-se de armazenar dados em colunas no disco, principalmente em termos de projeto de um banco de dados. A compressão de dados e as estratégias de materialização são algumas dessas características. A seguir serão abordadas e detalhadas essas

características.

Compressão de Dados

A Compressão de dados em sistemas de banco de dados traz a vantagem de diminuir a quantidade de acessos a disco. Essa diminuição é resultado da redução do tamanho total da massa de dados a ser transferida. Um problema resultante da utilização desta técnica seria o custo de processamento para realizar a descompressão dos dados (ABADI, 2006).

Em banco de dados orientados à coluna a compressão de dados tem a vantagem de poder operar normalmente sobre dados comprimidos (ABADI, 2006). Isso evita o desperdício de processamento para realizar a descompressão dos mesmos. Além desse fato, o armazenamento em coluna permite maiores taxas de compressão em função da similaridade (dados com a mesma natureza) entre os dados que são alocados sequencialmente no disco.

Ganhos significativos de performance podem ser obtidos por meio da utilização de compressão de dados. Além desses ganhos de performance, espaço em disco e memória podem ser poupados se bons esquemas de compressão forem utilizados. Sistemas de banco de dados que utilizam extensivamente replicação de dados deveriam empregar alguma técnica de compressão a fim de reduzir o espaço total de armazenamento necessário.

Estratégias de Materialização

Bancos de dados orientados à coluna modificam unicamente as estruturas no nível físico, enquanto os níveis lógico e de visões permanecem inalterados. Essa característica é necessária para que aplicações clientes que façam interface com um banco orientado à linha possam substituí-los pelo armazenamento em coluna, sem que haja necessidade de realizar adaptações para poderem continuar operando normalmente.

Sendo assim em algum momento após ser submetida uma consulta em um banco de dados orientado à coluna é necessário que os resultados, que nada mais

são, que várias colunas armazenadas independentemente, seja convertido em linhas. O problema é que o ponto exato em que essa conversão deve ocorrer não é tão óbvio (ABADI, 2007).

Nesse contexto, a técnica conhecida como materialização se refere ao fenômeno de reconstrução dos registros. As principais estratégias de materialização são (ABADI, 2007): Materializa ção Antecipada (*Early Materialization*) e a Materialização Tardia (*Late Materialization*). A seguir, serão abordadas essas estratégias que realizam a reconstrução dos registros a partir dos resultados intermediários resultantes das consultas submetidas em bancos de dados orientados à coluna.

- Materialização Antecipada: Esta estratégia de materialização impõe que os dados ao serem acessados sejam imediatamente adicionados em registros intermediários resultantes. Essa característica é especialmente útil em casos em que resultados intermediários devem ser constantemente re-acessados. Ao manter os registros intermediários construídos a Materialização Antecipada evita futuras reconstruções *desses dados.*

- Materialização Tardia: Nesta estratégia posterga-se o máximo possível no plano de uma consulta a reconstrução de um registro. Em alguns casos esta reconstrução só irá ocorrer no momento em que os resultados da consulta forem fornecidos. Os predicados de uma consulta normalmente reduzem a quantidade de registros resultantes e operações de agregação resumem em um atributo os resultados da mesma. Nesse sentido, a Materialização Tardia torna-se útil pois a mesma evita a reconstrução de registros intermediários desnecessários e reconstrói somente os registros relevantes. Esta estratégia de materialização permite também trabalhar com dados que estão comprimidos no disco em memória sem que haja a necessidade de descomprimi-los.

Apesar da Materialização Tardia prover diversas vantagens para a reconstrução dos registros (trabalhar com dados comprimidos e reconstruir somente os registros relevantes), o custo adicional de reprocessar registros intermediários em

alguns casos fazem com que a Materialização Antecipada seja preferível. Não existe consenso em qual momento a reconstrução dos registros deve ser realizada, dessa maneira é necessário empregar heurísticas para decidir qual das estratégias é mais útil em determinado caso.

No geral quando se irá trabalhar com consultas que possuem baixa seletividade, com dados comprimidos e/ou agregados a Materialização Tardia é a mais indicada, por outro lado, a Materialização Antecipada deve ser utilizada em casos de consultas que possuam alta seletividade, dados não comprimidos e sem agregações (ABADI, 2007).

2.3.3 PRINCIPAIS BANCOS DE DADOS QUE UTILIZAM ARMAZENAMENTO EM COLUNA

Em 1985 houve o lançamento do trabalho pioneiro no armazenamento em colunas denominado *A Decomposition Storage Model* (COPELAND, 1985). Após o lançamento da proposta original, na década de 90, foi colocado no mercado uma aplicação que explorava comercialmente os conceitos originados pelo DSM. Este produto se denominava *Sybase IQ* (MACNICOL, 2004) e era destinado ao uso em aplicações do tipo *Data Warehousing* e análises analíticas.

No final dos anos noventa um grupo de pesquisa desenvolveu o SGBD orientado à coluna MonetDB (BONCZ, 1999). Neste projeto foram realizadas análises detalhadas sobre o impacto das arquiteturas modernas de computadores, em especial no que diz respeito a utilização de memórias cache multi-nível com a intensão de aliviar o a grande diferença de velocidades entre a memória principal e a CPU utilizando o particionamento vertical das relações (MANEGOLD, 1999).

Após uma lacuna de 20 anos, com novos *hardwares* a disposição e realidades das aplicações bem diferentes, Daniel Abadi et al proveram o “renascimento” dos modelos orientados à coluna com uma nova abordagem que combinava diversas técnicas (STONEBRAKER, 2005). Esta nova abordagem ficou conhecida como C-Store e dentre as novas características importantes empregadas no novo sistemas pode-se citar processamento de dados comprimidos, otimização para a leitura, materialização tardia e o desenvolvimento de um motor de execução

de consultas que trabalha com os dados em coluna.

Após o surgimento do C-Store houve uma explosão de novos produtos, tanto comerciais quanto acadêmicos, que se utilizam do armazenamento em coluna, tais como, Vertica (*www.vertica.com*), LucidDB (www.luciddb.org), Infobright (www.infobright.com), Paraccel (www.paraccel.com), VectorWise (http://www.ingres.com/vectorwise/) entre outros.

2.4 TIPOS DE CARGAS DE TRABALHO EM BANCO DE DADOS

Comumente dividi-se as cargas de trabalho executadas em um banco de dados como sendo transacionais (*On Line Transaction Processing* - OLTP) ou analíticas (*On-Line Analitical Processing* - OLAP). Em geral, pode-se assumir que os sistemas OLTP tem o objetivo de prover uma fonte de dados, enquanto os sistemas OLAP tem a finalidade de analisá-los. A seguir serão abordados os principais fatos relativos a cada uma delas.

2.4.1 OLTP

As cargas de trabalho OLTP são caracterizadas por possuírem grandes quantidades de transações curtas e em tempo real, que são executadas várias vezes em um determinado período de tempo (WREMBEL, 2007). Como exemplo deste tipo de carga pode-se citar as inserções, atualizações e exclusões. A principal característica das consultas OLTP é que as mesmas são muito rápidas, e normalmente são executadas em um ambiente multiacesso. A eficiência deste tipo de consulta é medida em transações por segundo.

Banco de dados que fazem uso extensivo desse tipo de consulta possuem dados atualizados e utilizam esquemas normalizados para armazená-los. As consultas OLTP normalmente são instantâneas e processadas milhares de vezes por dia, onde cada uma delas irá conter uma pequena porção de dados.

2.4.2 OLAP

As cargas OLAP são caracterizadas por possuírem uma quantidade relativamente baixa de transações. Neste tipo de carga de trabalho são executadas consultas que são normalmente muito complexas, sobre extensas massas de dados e que frequentemente envolvem diversas formas de agregações entre eles (WREMBEL, 2007). Aplicações OLAP são amplamente utilizados em *Data Warehouses*, em *Business Intelligence* e por técnicas de *Data Mining*.

Sistemas que utilizam esse tipo de carga de trabalho possuem bases de dados volumosas, que contém dados históricos e que são armazenados em esquemas multidimensionais. A tabela 1.1 resume as principais características e diferenças entre os dois tipos de carga de trabalho OLAP e OLTP.

Tabela 2.1: Resumo das Características da Consultas OLAP e OLTP

	OLTP	OLAP
Tipo e fonte de dados.	Dados operacionais. OLTP são fonte de dados.	Dados consolidados que apresentam uma visão multidimensional de vários tipos de negócios. Os dados OLAP vem de vários bancos transacionais.
Finalidade dos dados	Para controlar e executar tarefas fundamentais do negócio.	Para ajudar no planejamento, resolução de problemas, e apoio a decisão.
Inserções e Atualizações	Inserções e atualizações curtas rápidas e simples, que são iniciadas por usuários finais.	Trabalhos em lotes periódicos de longa duração para inserir e atualizar os dados.
Consultas	Consultas relativamente padronizadas e simples	Normalmente consultas complexas envolvendo

	retornando normalmente poucos registros.	várias agregações.
Velocidade de processamento	Normalmente muito rápido.	Depende da quantidade de dados envolvidos; Consultas complexas podem levar várias horas.
Requisitos de Espaço	Relativamente pequeno.	Normalmente grande.

FONTE: WREMBEL (2007)

3 METODOLOGIA

3.1 METODOLOGIA

A metodologia de desenvolvimento adotada na construção do trabalho proposto foi a *Rational Unified Process* (RUP). Uma metodologia para o desenvolvimento de um software deve especificar uma sequência de passos a ser seguidos durante o seu desenvolvimento. A seção seguinte irá detalhar esta metodologia.

3.2 MODELO DE PROCESSO DE ENGENHARIA DE SOFTWARE

O processo de desenvolvimento unicado da Rational (RATIONAL, 2012) consiste em um modelo de desenvolvimento de software interativo e incremental, o qual é constituído por duas dimensões. Essas dimensões podem ser visualizadas por meio da figura 3.1.

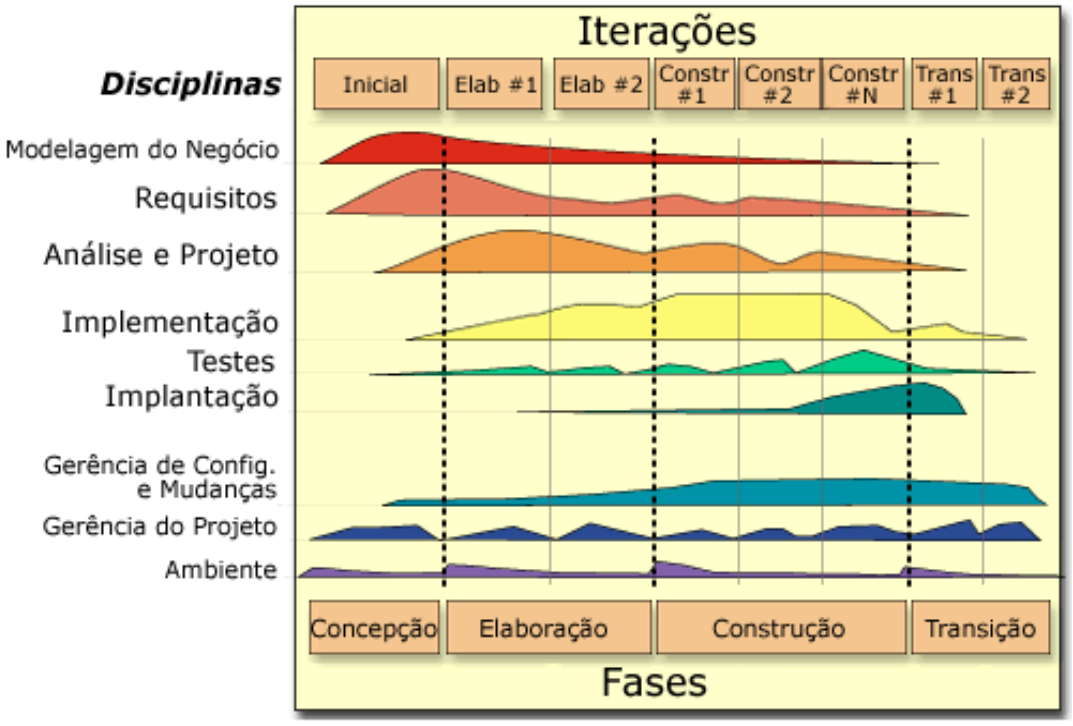


Figura 3.1 – RUP
FONTE: RATIONAL (2012).

O eixo vertical do modelo representa aspectos estáticos do processo e é composto por disciplinas. Cada disciplina agrupa de maneira lógica várias atividades que estão naturalmente relacionadas.

O eixo horizontal representa aspectos dinâmicos do processo de desenvolvimento, nele se encontram elementos relacionados ao tempo, os quais são denominados fases. O RUP possui quatro fases, que são a Concepção, Elaboração, Construção e Transição. Cada fase pode ter uma ou mais iterações, e cada iteração irá representar um ciclo de desenvolvimento completo (ou seja, irá gerar um *release* do produto). O m de uma fase é caracterizado por um *milestone,* que é um marco que indica a conclusão de um conjunto de tarefas pertencentes à fase. Em cada *milestone* são gerados alguns artefatos, como diagramas, documentos e o próprio código fonte (RATIONAL, 2012).

Os dados que estão plotados no gráfico representam os esforços relativos a cada disciplina em determinada fase do ciclo de desenvolvimento. Pode-se observar que algumas disciplinas possuem poucas contribuições em algumas fases, e se fazem extremamente necessárias em outras.

O ciclo de vida iterativo do RUP apresenta várias vantagens. Erros pertencentes a fases de análise e design podem ser descobertos e corrigidos antecipadamente, pois ao contrário de outros processos de desenvolvimento (como o cascata), o RUP aplica testes a cada iteração desenvolvida, e não somente ao final da fase de construção.

3.3 PLANO DE ATIVIDADE

O plano de atividade é composto pelo diagrama WBS (Work Breakdown Structure) e o gráfico de GANTT. A seguir são apresentados cada um desses artefatos.

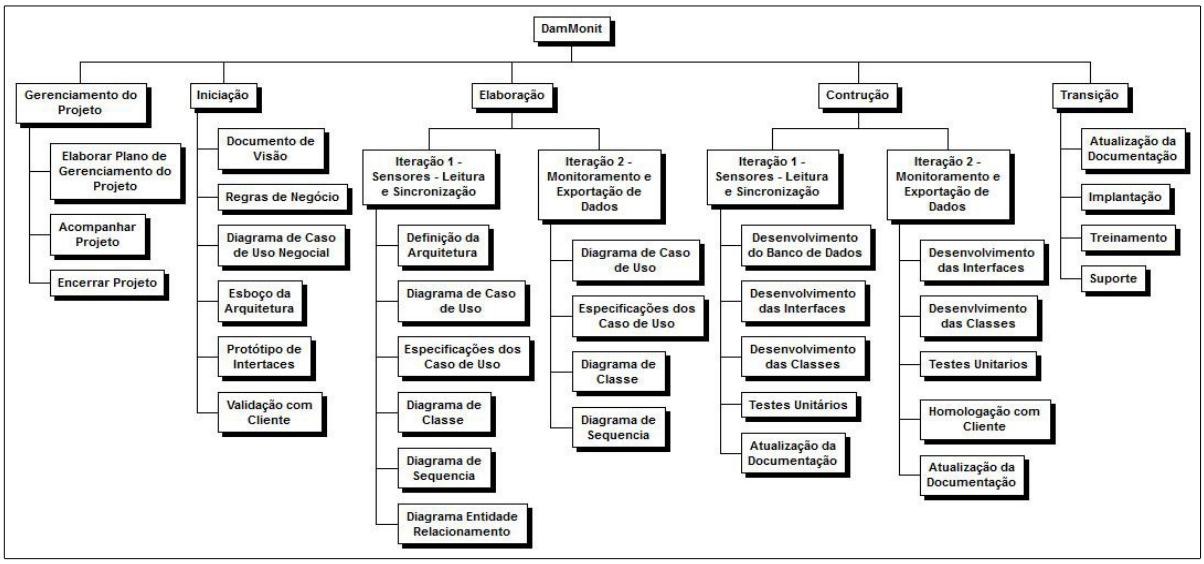


Figura 3.1 – WBS. FONTE: O autor (2012).

A figura 3.1 apresenta a WBS, cuja função é dar uma visão geral do projeto, mostrando cada uma de suas fases juntamente com as atividades que compõem cada uma delas

		Nome	Duração	Início	Trabalho	Término
1		**Dam Monit**	**59,25 dias?**	**10/09/12 19:00**	**271,5 horas**	**16/12/12 20:00**
2		**Gerenciamento do Projeto**	**59,25 dias?**	**10/09/12 19:00**	**34,5 horas**	**16/12/12 20:00**
3		Elaborar Plano de Gerenciamento de Projeto	2 dias	10/09/12 19:00	8 horas	15/09/12 10:00
4		Acompanhar Projeto	56,25 dias	15/09/12 10:00	22,5 horas	15/12/12 14:30
5		Encerrar Projeto	1 dia?	16/12/12 16:00	4 horas	16/12/12 20:00
6		**Iniciação**	**6 dias?**	**15/09/12 10:00**	**24 horas**	**23/09/12 19:00**
7		Documento de Visão	1 dia?	15/09/12 10:00	4 horas	15/09/12 14:30
8		Regras de Negócio	0,5 dias?	16/09/12 16:00	2 horas	16/09/12 18:00
9		Diagrama de Caso de Uso Negocial	0,5 dias?	16/09/12 18:00	2 horas	16/09/12 20:00
10		Esboço da Arquitetura	1 dia?	16/09/12 20:00	4 horas	17/09/12 22:00
11		Protótipo de Interfaces	2 dias?	21/09/12 19:00	8 horas	22/09/12 13:30
12		Validação com o Cliente	1 dia	22/09/12 13:30	4 horas	23/09/12 19:00
13		**Elaboração**	**11 dias?**	**23/09/12 19:00**	**52 horas**	**13/10/12 10:00**
14		**Iteração 1 - Sensores - Leitura e Sincronização**	**6 dias?**	**23/09/12 19:00**	**32 horas**	**05/10/12 21:00**
15		Definição da Arquitetura	2 dias?	23/09/12 19:00	8 horas	28/09/12 22:00
16		Diagrama de Caso de Uso	0,5 dias?	23/09/12 19:00	2 horas	23/09/12 21:00
17		Especificações dos Caso de Uso	1,5 dias?	24/09/12 19:00	6 horas	28/09/12 22:00
18		Diagrama de Classes	1 dia?	29/09/12 08:00	4 horas	29/09/12 12:30
19		Diagrama de Sequencia	2 dias?	29/09/12 12:30	8 horas	01/10/12 20:00
20		Diagrama Entidade Relacionamento	1 dia?	01/10/12 20:00	4 horas	05/10/12 21:00
21		**Iteração 2 - Monitoramento e Exportação de Dados**	**5 dias?**	**05/10/12 21:00**	**20 horas**	**13/10/12 10:00**
22		Diagrama de Caso de Uso	0,5 dias?	05/10/12 21:00	2 horas	06/10/12 09:00
23		Especificações dos Caso de Uso	1,5 dias?	06/10/12 09:00	6 horas	07/10/12 17:00
24		Diagrama de Classes	1 dia?	07/10/12 17:00	4 horas	07/10/12 21:00
25		Diagrama de Sequencia	2 dias?	08/10/12 19:00	8 horas	13/10/12 10:00
26		**Construção**	**35,25 dias?**	**13/10/12 10:00**	**141 horas**	**09/12/12 17:00**
27		**Iteração 1 - Sensores - Leitura e Sincronização**	**17,25 dias?**	**13/10/12 10:00**	**69 horas**	**10/11/12 11:00**
28		Desenvolvimento do Banco de Dados	1 dia?	13/10/12 10:00	4 horas	13/10/12 14:30
29		Desenvolvimento das Interfaces	4,75 dias?	14/10/12 16:00	19 horas	21/10/12 18:00
30		Desenvolvimento das Classes	9,5 dias?	21/10/12 18:00	38 horas	05/11/12 20:00
31		Testes Unitarios	1 dia?	05/11/12 20:00	4 horas	09/11/12 21:00
32		Atualização da Documentação	1 dia?	09/11/12 21:00	4 horas	10/11/12 11:00
33		**Iteração 2 - Monitoramento e Exportação de Dados**	**18 dias?**	**10/11/12 11:00**	**72 horas**	**09/12/12 17:00**
34		Desenvolvimento das Interfaces	3 dias?	10/11/12 11:00	12 horas	16/11/12 20:00

DamMonit - página1

Figura 3.2 – Gantt. FONTE: O autor (2012).

A imagem apresentada na figura 3.2 apresenta o gráfico de Gantt, cuja importância é descrever o cronograma de execução do projeto, listando o prazo de conclusão de cada uma das atividades do projeto e suas respectivas dependências.

3.4 PLANO DE RISCOS

O projeto DamMonit apresenta dois riscos potenciais, sendo um relacionado ao projeto do sistema e o outro a construção das interfaces. O plano de contingência para ambos os casos é realizar estudos aprofundados em relação ao problema a ser resolvido (para sanar a condição número 1) e possibilitar um aprendizado profundo aos desenvolvedores dos componentes PrimeFaces para confecção das interfaces (condição número 2). O Plano de Riscos está ilustrado na tabela 3.1.

Tabela 3.1 – Plano de Riscos

N°	**Condição**	**Data Limite**	**Consequência**	**Ação**	**Monitoramento**	**Prob**	**Impacto**	**Classe**
1	Dificuldade na modelagem da base de dados híbrida	20/11	Atrasos na análise, inviabilidade do projeto	Considerar Contingência de tempo nas atividades	Feedback Baixa atrevés de reuniões diárias.	Baixo	Muito Alto	6
2	Deficiência da equipe em desenvolvimento de interfaces gráficas mais elaboradas	30/11	Atrasos na implementação, possível aumento de custos do projeto.	Considerar Contingência de tempo nas atividades	Feedback Médio através de reuniões diárias.	Médio	Alto	6

FONTE: O autor (2012).

3.5 RESPONSABILIDADES

Abaixo estão listadas os principais participantes e suas respectivas responsabilidades no desenvolvimento do projeto:

- Gerente do Projeto: Acompanha o andamento de todo o projeto, assumindo a responsabilidade de mitigar possíveis riscos de insucesso do projeto;

- Analista de Sistemas: Desenvolve o projeto do sistema, gerando artefatos que serão utilizados na confecção do produto final;

- Programador: Implementa os casos de uso descritos na documentação, transformando-os no produto final;

- Tester: Testa as funcionalidades desenvolvidas e verifica se as mesmas atendem aos requisitos descritos na documentação do sistema.

3.6 MATERIAIS

Para implementar a solução proposta, forma utilizadas as seguintes tecnologias:

- Metodologia de desenvolvimento RUP;
- Linguagem de programação Java;
- Suíte de componentes de interface Primefaces;
- Servidor web GlassFish 3.1.2;
- Banco de dados MySql 5.2.
- Ferramenta Case Astah.

O hardware utilizado para a construção do software foi um notebook com a seguinte configuração: Intel Core I5 2.5 Ghz, 4 Gb de memória RAM e 500 Gb de HD.

3.7 ESTUDO DE CASO

Como visto anteriormente, a performance das consultas a serem submetidas contra o banco de dados depende fortemente do tipo de armazenamento físico no qual os dados a serem processados se encontram em disco. Dessa maneira, o principal objetivo desta arquitetura híbrida é favorecer qualquer que seja a carga de trabalho a ser submetida ao banco, independente de sua natureza, sendo assim o sistema a ser desenvolvido irá utilizar uma arquitetura híbrida de armazenamento de dados, parte relacional parte orientada a coluna.

As características do sistema a ser desenvolvido evidenciam a necessidade de garantir uma alta performance tanto para cargas de trabalho analíticas (monitoramento dos dados dos sensores da instrumentação) quanto para cargas de trabalho transacional (cadastro de usuários, instrumentos e etc).

Se os objetivos pretendidos forem alcançados a arquitetura proposta será

uma alternativa viável, tanto para sistemas que envolvam cargas de trabalhos mistas. A seguir será apresentado o estudo de caso que irá viabilizar o estudo da arquitetura híbrida proposta.

3.7.1 A SEGURANÇA DE UMA BARRAGEM HIDRELÉTRICA

Barragens são empreendimentos complexos que criam obstáculos que impedem o fluxo natural da águas e têm diversas finalidades, tais como: acúmulo de água para a irrigação e abastecimento, formação de lagos artificiais para a navegação e geração de energia (ICOLD, 2012).

Estima-se que o Brasil possua mais de 2000 barragens de médio e grande porte atualmente, o que caracteriza a utilização de apenas um terço de seu potencial hidrológico para a geração de energia elétrica. O bom funcionamento desses empreendimentos, em todos seus aspectos, é vital para que o país possa aproveitar todo o seu potencial hídrico para a produção de energia elétrica (CBDB, 1996).

Dentro desse contexto a segurança de uma barragens tem grande importância, pois ela visa mantê-las em bom estado, afim de evitar perdas materiais e de vidas humanas em função de possíveis rupturas e outros problemas associados à essas estruturas.

3.7.2 SENSORES DE BARRAGENS HIDRELÉTRICAS

A instrumentação representa um dos aspectos mais importantes quando se refere a segurança de barragens. É por meio dos dados fornecidos a partir das leituras realizadas nos instrumentos instalados nas barragens que é possível realizar análises a respeito das condições em que a mesma se encontra, além de poder fazer detecções precoces de problemas em potencial que a barragem pode apresentar ao longo do tempo. Leituras de instrumentos realizadas de forma correta e em épocas apropriadas são fundamentais para determinar o desempenho e a segurança de uma barragem (KUPERMAN, 2003).

Existem uma série de parâmetros que podem ser monitorados em uma barragem, tais como, a pressão da água nos poros, a pressão da água na rocha de

fundação, as pressões totais, as cargas e as tensões nos elementos estruturais, a temperatura, as vazões de drenagem, e os materiais sólidos careados (SILVEIRA, 2006). A medição das grandezas monitoradas fornece indicativos de causas e efeitos, e com base nesses indicativos pode-se construir relações entre esses parâmetros. Essas relações irão permitir que análises sejam realizadas de modo que sejam previstas e remediadas situações indesejadas que possam
surgir ao longo do tempo (SILVEIRA, 2006).

3.7.3 DESCRIÇÃO DA PROBLEMÁTICA

As leituras realizadas pelos instrumentos de uma hidrelétrica geram uma quantidade maciça de dados. Cada um dos instrumentos instalados em sua barragem, cuja quantidade pode chegar aos milhares, realiza leituras periódicas que podem variam de mês a mês até segundo a segundo.

Dado essa enorme quantidade informações coletadas e a necessidade de monitoramento contante a fim de garantir a segurança da barragem de uma hidrelétrica é necessário a construção de uma aplicação que viabilize o monitoramento com boa performance e que seja disponibilizado de forma *on-line*. Além do monitoramento o sistema deve ser capaz de realizar o cadastro de usuários, instrumentos, e locais da hidrelétrica.

3.7.4 SOLUÇÃO PROPOSTA

Para resolver a problemática descrita na sessão anterior será implementado um sistema web cuja com uma base de dados híbrida, parte orientada a linha, parte orientada a coluna. Esta a escolha desta organização tem como objetivo maximizar a performance das consultas realizadas na base independente das características das mesmas. Na figura abaixo é apresentado o esquema geral desta arquitetura híbrida.

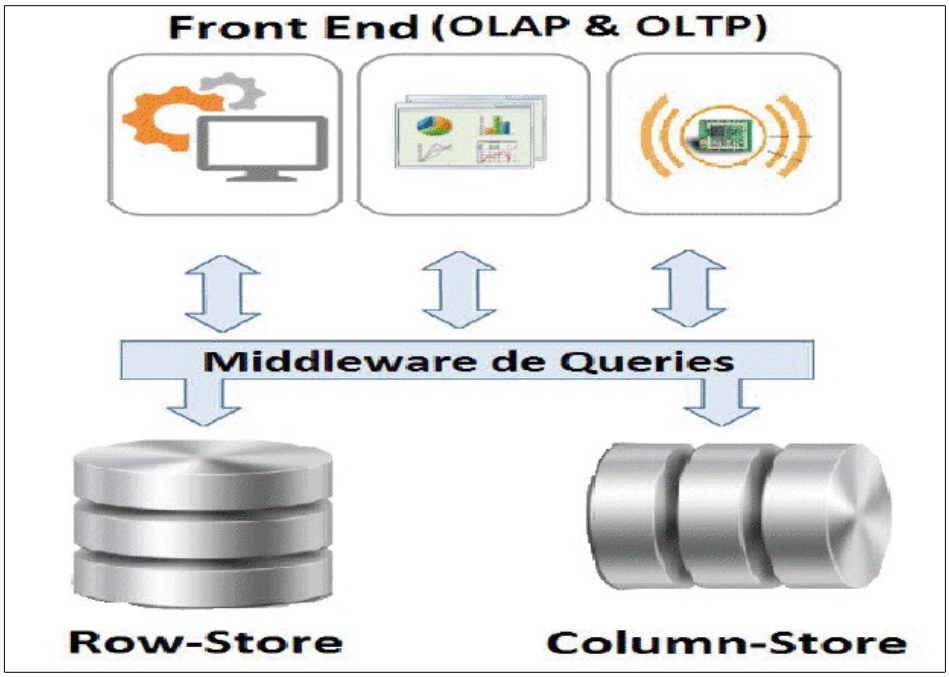


Figura 3.3 – Esquema da Arquitetura híbrida Linha-Coluna
FONTE: O autor (2012).

4 APRESENTAÇÃO DO SOFTWARE

Neste capítulo será apresentado o software de monitoramento de instrumentação de barragens DamMonit. Para facilitar a explicação das funcionalidades do sistema as mesmas foram separadas em módulos. Neste capítulo também é realizada a discussão dos testes realizados, sendo que estes são de grande importância para comprovar a eficiência da arquitetura híbrida proposta.

4.1 MÓDULOS DO SISTEMA

1) Módulo de Administração: Este módulo concentrará todas as atividades necessárias à manutenção do sistema, compreendendo o cadastro dos usuários que irão utilizar o sistema, cadastro dos sensores instalados na barragem, e cadastro das localizações geográficas da barragem. As figuras a seguir ilustram, respectivamente o cadastro e listagem de usuários e cadastro de instrumentos.

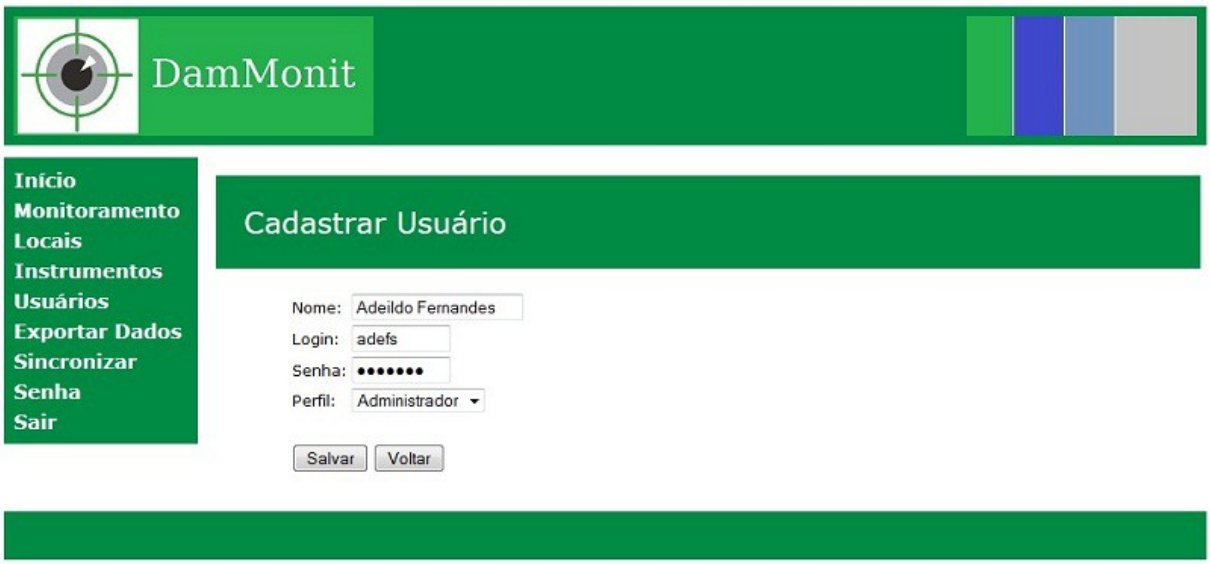


Figura 4.1 – Cadastro de Usuários.
FONTE: O autor (2012).

Figura 4.2 – Listagem de Usuários.
FONTE: O autor (2012).

Figura 4.3 – Cadastro de Instrumentos.
FONTE: O autor (2012).

2) Módulo de Monitoramento: Este módulo será o responsável por exibir os resultados obtidos a partir das leituras realizadas pelos sensores. Neste módulo o usuário poderá submeter ao sistema diversos tipos de consultas a fim de acompanhar o estado de seguridade da barragem. A principal tabela da base de dados que sustenta este módulo é a tabela 'leitura'.

É nesta relação que ocorre a hibridização de modelos de armazenamento de dados, ou seja, a mesma é mantida no modelo clássico relacional juntamente com suas partições verticais (modelo orientado a coluna). Esta estratégia foi adotada devido a grande quantidade de dados que a mesma possui e a necessidade de manter uma boa performance de acesso. Na figura 4.4 é ilustrada a tela do sistema responsável pela realização das pesquisas.

Figura 4.4 – Tela de Monitoramento do Sistema.
FONTE: O autor (2012).

3) Módulo de Exportação de Dados: Responsável por realizar a exportação dos dados captados para um formato específico (.arff) utilizado pelo programa de mineração de dados *weka* (http://www.cs.waikato.ac.nz/ml/weka/). A mineração dos dados obtidos a partir dos leitores pode enriquecer os estudos em segurança de barragens, sendo assim este módulo agrega grande valor ao sistema. A seguir a tela de exportação de dados é ilustrada na Figura 4.4.

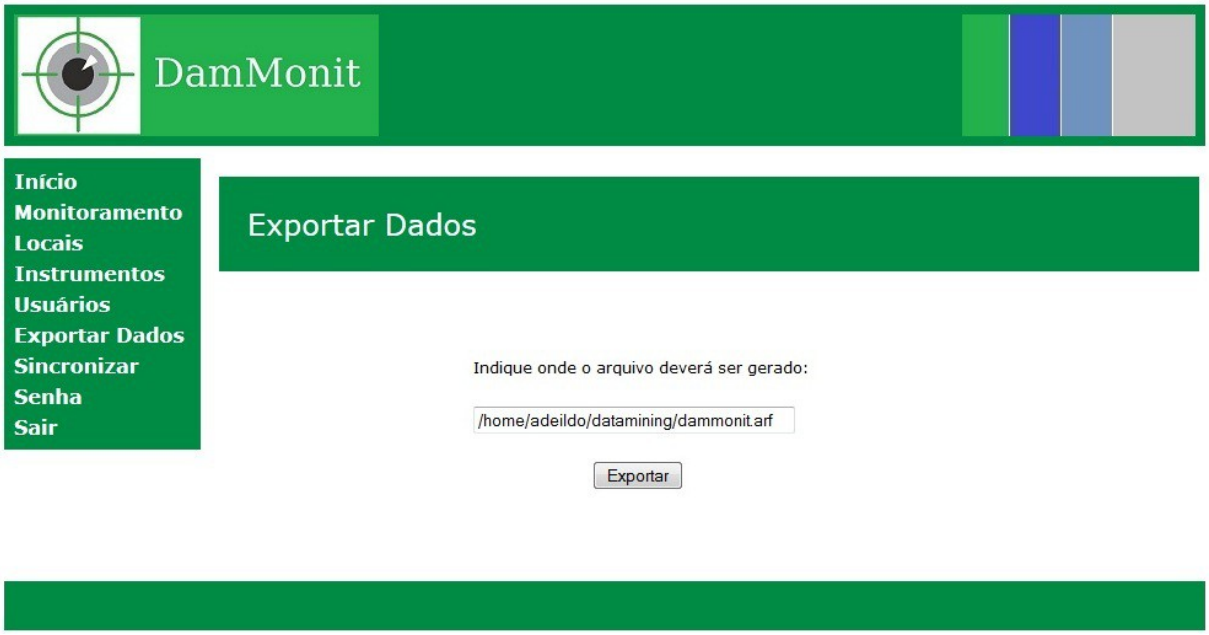


Figura 4.4 – Exportar Dados.
FONTE: O autor (2012).

4) Módulo de Sistema: Este módulo compreenderá rotinas automáticas realizadas pelo sistema, sendo elas a leitura dos dados produzidos pelos sensores a partir de arquivos locais e sincronização dos dados entre os esquemas orientados a linha e orientados a coluna. Devido as rotinas serem executadas automaticamente pelo sistema as mesmas não possuem telas.

4.2 INSTALAÇÃO DO SISTEMA

Para realizar a instalação do sistema deverão ser realizados uma série de passos, listados a seguir, que deverão ser rigorosamente seguidos. Todos os softwares necessários, assim como os fontes, se encontram no DVD anexado a este trabalho. Cronologicamente as instruções a serem realizadas são:

1) Instalação do servidor *web* GlassFish;
2) Instalação do banco de dados MySQL (usuário 'Admin', senha '12345');
3) Instalação do *browser* Firefox;
4) Criar as tabelas do banco executando o script DamMonit.sql;
5) Realizar o deploy do sistema com o arquivo WAR.
6) Iniciar o servidor web.

Após realizar o *Start* no servidor *web,* basta abrir um *browser* e acessar o endereço: http://localhost:8080/dammonit para acessar a aplicação que já se encontra pronta para ser utilizada. Um usuário padrão para o primeiro acesso ao sistema foi criado com o login: Admin e a senha: 12345.

4.3 TESTES DE PERFORMANCE

Esta sessão irá apresentar os testes realizados para verificar a eficiência da arquitetura híbrida proposta. Os *scripts* para a criação da base de dados que foi utilizada nos testes se encontra no DVD anexo ao trabalho.

Para a realização dos experimentos populou-se a tabela 'leituras' (do esquema *row-store*) com 10.000.000 de registros, sendo que os valores das colunas foram gerados aleatoriamente. Posteriormente populou-se, de maneira consistente com o esquema row-store, as relações do esquema *column-store* (que foram obtidas a partir do particionamento vertical da relação leituras). O tamanho total da base de dados foi de 12.600 Gb.

Após a construção da base de dados, foram definidas quais seriam as cargas de trabalho a serem executadas contra a base. Metade das consultas realizadas tinham a característica transacional e a outra metade analítica. As consultas utilizadas para os experimentos (seguidas dos SQL do esquema *row-store* e *column-store* respectivamente) são listadas abaixo:

01 – [T] Selecione todas as leituras realizadas no ano de 1986 pelo instrumento 1.
[R] → SELECT * FROM leitura WHERE idInstrumento = 1 AND ano = 1986
[C] → SELECT leitura_instrumento.value, leitura_temperatura.value, leitura_dia.value, leitura_mes.value, leitura_ano.value, leitura_minuto.value, leitura.segundo.value, leitura_hora.value, leitura_1.value, leitura_2.value, leitura_3.value, leitura_4.value, leitura_5.value FROM leitura_instrumento INNER JOIN leitura_temperatura INNER JOIN leitura_dia INNER JOIN leitura_mes INNER JOIN leitura_ano INNER JOIN leitura_minuto INNER JOIN leitura_segundo INNER JOIN leitura_hora INNER JOIN leitura_1 INNER JOIN leitura_2 INNER JOIN leitura_3 INNER JOIN leitura_4 INNER JOIN leitura_5 WHERE leitura_ano.value 1986 AND leitura_instrumento = 1;

02 – [T] Insira uma nova leitura
[R] → INSERT INTO leitura (idInstrumento, temperatura, dia, mês, ano, minuto, segundo, hora, leitura_1, leitura_2, leitura_3, leitura_4, leitura_5,) VALUES (1,32,1,3,2012,30,23,15,1.22,3.456,5.09,7.8,0.03);
[C] → INSERT INTO leitura_instrumento (value) VALUES (1); INSERT INTO leitura_temperatura (value) VALUES (32); INSERT INTO leitura_dia (value) VALUES (1); INSERT INTO leitura_mes (value) VALUES (3); INSERT INTO leitura_ano (value) VALUES (2012); INSERT INTO leitura_minuto (value) VALUES (30); INSERT INTO leitura_segundo (value)

VALUES (23); INSERT INTO leitura_hora (value) VALUES (15); INSERT INTO leitura_1 (value) VALUES (1.22); INSERT INTO leitura_2 (value) VALUES (3.456); INSERT INTO leitura_3 (value) VALUES (5.09); INSERT INTO leitura_4 (value) VALUES (7.8); INSERT INTO leitura_5 (value) VALUES (0.03);

03 – [T] Remova todas as leituras cuja temperatura é superior a 100
[R] → DELETE FROM leitura WHERE temperatura > 100;
[C] → DELETE FROM leituras WHERE temperatura > 100;
DELETE FROM leitura_instumento INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_dia INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_instumento INNER JOIN leitura_mes WHERE leitura.temperatura.value > 100; DELETE FROM leitura_ano INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_minuto INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_segundo INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_hora INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_1 INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_2 INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_3 INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_4 INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_5 INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100; DELETE FROM leitura_temperatura INNER JOIN leitura_temperatura WHERE leitura_temperatura.value > 100;

04 – [T] Altere para todas as leituras o valor de leitura 5 para 0
[R] → UPDATE leituras SET leitura_5 = 0;
[C] → UPDATE leitura_5 SET value = 0;

05 – [T] Selecione todas as leituras do tipo 4 cuja temperatura está entre 25 e 28 graus
[R] → SELECT leitura_4 FROM leitura WHERE temperatura BETWEEN 25 AND 28;
[C] → SELECT leitura_4.value FROM leitura_4 INNER JOIN leitura_temperatura WHERE leitura_temperatura.value BETWEEN 25 AND 28;

06 – [A] Selecione a média de todas as leituras do tipo 1
[R] → SELECT AVG(leitura1) FROM leituras;
[C] → SELECT AVG(value) FROM leitura_1;

07 – [A] Selecione a temperatura máxima realizada em uma leitura
[R] → SELECT MAX(temperatura) FROM leituras;
[C] → SELECT MAX(value) FROM temperatura;

08 – [A] Selecione a temperatura mínima do ano de 1980
[R] → SELECT MIN(temperatura) FROM leitura WHERE ano = 1980;
[C] → SELECT MIN(value) FROM leitura_temperatura INNER JOIN leitura_ano WHERE leitura_ano.value = 1980;

09 – [A] Conte a ocorrência de leitura do tipo 2 do mês de dezembro de 1995 maiores que 5.0
[R] → SELECT COUNT(*) FROM leitura WHERE mês = 12 AND ano = 1995 AND leitura2 >5.0;
[C] → SELECT COUNT(*) FROM leitura_2 INNER JOIN leitura_ano INNER JOIN leitura_mes WHERE leitura_ano.value = 1995 AND leitura_mes.value = 12 AND leitura_2.value > 5.0 ;

10 – [A] Conte quantas leituras o sistema já registrou
[R] → SELECT COUNT(*) FROM leitura;
[C] → SELECT COUNT(*) FROM leitura_instrumento;

O propósito do experimento foi capturar o tempo de execução (em milissegundos) de cada consulta em ambos os esquemas de armazenamento. Os resultados obtidos, das consultas detalhadas acima pode ser visualizado na tabela 4.1. Os resultados descritos na tabela são resultado da média de 10 (dez) execuções de cada uma das consultas.

Tabela 4.1 – Resultado do Experimento

Transacional	**01**	**02**	**03**	**04**	**05**
Row-Store	**23,84**	**0,094**	**84,475**	**44,851**	**12,137**
Column-Store	**768,73**	**1,187**	**323,063**	**8.33**	**5,98**
Analitíco	**06**	**07**	**08**	**09**	**10**
Row-Store	**10,202**	**9,625**	**10.124**	**10,62**	**5.288**
Column-Store	**5,912**	**5,164**	**5.569**	**3.853**	**3.978**

FONTE: O autor (2012).

Como pode-se pela tabela 4.1 em geral a base orientada a linha (Row-Store) teve um melhor desempenho com consultas transacionais e a base orientada a coluna (Column-Store) nas consultas analíticas.

Para as consultas analíticas o modelo C-Store teve um rendimento cerca de duas vezes melhor do que o modelo Row-Store, chegando a ser cerca de três vezes mais eficiente no caso da consulta número 09.

As consultas 01 e 03 mostram a total ineficiência do modelo Column-Store quando é necessário realizar operações nas quais a manipulação de todos os atributos é necessária. Fica evidente que para esses casos a melhor opção é o modelo Row-Store.

As consultas 04 e 05 embora tenham a característica de serem transacionais tiveram melhor desempenho quando executadas na base Row-Store. Essa exceção se deveu ao fato destas consultas acessarem poucos atributos da tabela “leitura”. A medida que fosse aumentado a quantidade de atributos acessados na tabela é provável que o desempenho do modelo Column-Store seria empobrecido, fazendo com que o modelo Row-Store fosse o mais adequado para este caso.

5 CONSIDERAÇÕES FINAIS

Os atuais sistemas de informação fazem uso extensivo de Banco de Dados para armazenar de forma persistente suas informações, sendo assim um banco de dados é essencial para as mais diversas aplicações desenvolvidas. Atualmente a maior parte dos SGBDs agrupa todos os atributos de um registro e os armazena sequencialmente em disco. Este tipo de arquitetura, possibilita que a execução de algumas operações tenha alto desempenho em função da disposição em que os dados se encontram armazenados.

Operações como inserções, exclusões e frequentemente atualizações tem que acessar todos os atributos de um registro para realizar seus processamentos. A disposição sequencial dos dados em disco permite que esses tipos de operações sejam eficientes, pois um único acesso a disco é necessário para recuperar um registro. Em função dessa característica sistemas orientados a linha frequentemente são chamados de “otimizados para escrita”, o que é especialmente relevante em consultas do tipo transacional.

O modelo relacional (orientado a linha) tem sido o modelo de armazenamento que tem prevalecido nas últimas décadas. No entanto, com o passar dos anos mudanças tanto arquiteturais quanto nos requisitos das aplicações fizeram surgir a necessidade de mudanças nesse paradigma de armazenamento (AILAMAKI, 1999). Diante deste cenário diversos cientistas da área de banco de dados perceberam que o então atual modelo de armazenamento padrão não era capaz de suprir a necessidade das novas arquiteturas e aplicações que rapidamente emergiam.

Isto fez com que diversas pesquisas com a intensão de resolver esse problema fossem desenvolvidas ao longo dos anos. Essas pesquisas resultaram na criação de uma novo modelo de armazenamento que ficou conhecido como modelo de armazenamento orientado à coluna (COPELAND, 1985).

Este novo modelo armazena cada atributo de uma relação (do ponto de vista lógico) separadamente e coloca seus valores sequencialmente em disco. Isto permite que cada atributo possa ser acessado de maneira independente quando requisitado. Em função da maior parte das consultas acessarem somente uma

pequena fração dos atributos de um registro faz com que o armazenamento em coluna, devido a possibilidade de acessá-los individualmente, resulte em uma menor quantidade de acessos a disco e consequentemente maiores taxas de E/S (COPELAND, 1987).

Isto faz com que este modelo seja “otimizado para leitura”, o que beneficia evidentemente operações do tipo analítica. É importante destacar que embora o modelo de armazenamento em coluna tenha adicionado bom desempenho para consultas orientadas à leitura, o mesmo não pode ser dito com relação as operações de escrita. Na prática uma consulta que necessitar realizar escritas em uma relação que contém *n* atributos implica realizar *n* acessos a disco, fazendo com que o custo da execução de inserções, exclusões e atualizações neste modelo se torne proibitivo.

A partir das observações e experimentos realizados pode-se concluir que o tipo de armazenamento, juntamente com as características das consultas influenciam fortemente na performance das mesmas. Tanto o armazenamento em linha quanto o armazenamento em coluna trazem grandes benefícios para determinados tipos de consultas em detrimento de outras.

Sistemas orientados à linha embora sejam otimizados para escrita, com o passar dos anos e com o aumento exponencial dos dados, tem se tornado ineficientes para consultas que fazem acesso a milhares de registros. Por outro lado, sistemas orientados a coluna acessam eficientemente grandes volumes de dados, porém são altamente ineficientes quando o assunto é escrita em disco (HALVERSON, 2006). Sendo assim, a combinação de ambos os esquemas de armazenamento, se utilizado de maneira correta, possibilita a melhora de performance global de um sistema que possui cargas de trabalho mistas.

REFERÊNCIAS

ABADI, D.J. **Column-Stores For Wide and Sparse Data CIDR***, California, USA,* 2006

ABADI, D.J., DEWITT, D.J. **Materialization strategies in a column-oriented dbms**. *ICDE*, Istanbul, Turkey, 2007.

ABADI, D. J., MADDEN, S. R. **Integrating compression and execution in column-oriented database systems**. *SIGMOD*, páginas 671–682, Chicago, IL, USA, 2006.

ABADI, D.J. **Query execution in column-oriented database systems**. MIT PhD Dissertation, 2008. PhD Thesis.

AILAMAKI, A. DEWITT, D. J. **Wood DBMSs On A Modern Processor: Where Does Time Go?** *VLDB Conference*, Edinburgh, Scotland, 1999.

BONCZ, P. A., KERSTEN, M. L. **MIL primitives for querying a fragmented world.** *VLDB Journal: Very Large Data Bases*, páginas 101–119, 1999.

BONCZ, P., ZUKOWSKI, M. **Monetdb/x100: Hyper-pipelining query execution.** *CIDR*, Amsterdam, The Netherlands, 2005.

CBDB, C. B. de B. **Auscultação e instrumentação de barragens no brasil. In: de Instrumentação de Barragens.** Belo Horizonte-MG: [s.n.], 1996.

COPELAND, G.P. , KHOSHAFIAN S. F. **A decomposition storage model.** *In Proceedings ACM Sigmod Conference on Management of Data*, Austin, Texas, 1985.

COPELAND, G.P. , KHOSHAFIAN S. F, et al. **A query processing strategy for the decomposed storage model.** *ICDE*, páginas 636–643, 1987.

ICOLD, **Comitê Internacional de Grandes Barragens**. 2008. Disponível em: <http://www.icold-cigb.org>. Acesso em: 28 ago. 2012.

HALVERSON, A., BECKMANN, J. L. et al. **Comparison of C-Store and Row-Store in a Common Framework** *32nd VLDB Conference*, Seoul, Korea, 2006.

KUPERMAN, S. C. et al. **Critérios para fixação de valores limites da instrumentação civil de barragens de concreto e de terra**. In: Salvador BA. In: XXV Seminário de Grandes Barragens - [S.l:.s.n.], 2003. p. 81-96.

MADDEN, S. R., HACHEM N. **Column-stores vs. Row-stores: How different are they really?** *SIGMOD*, Vancouver, Canada, 2008.

MANEGOLD, S. **Database Architecture Optimized for the new Bottleneck: Memory Access.** *VLDB*, Edinburgh, Scotland, 1999,

MANEGOLD, S., KERSTEN, M. **Optimizing database architecture for the new bottleneck: memory access.** *VLDB Journal*, páginas 231–246, Amsterdam, The Netherlands, 2000.

MACNICOL, R., FRENCH, B. **Sybase IQ multiplex - designed for analytics.** *VLDB* páginas 1227–1230, 2004.

RAMAKRISHNAN, R. **Database Management Systems**. WCB/McGraw-Hill, 2 edição, 2000.

RATIONAL. **Documentação RUP**. 2012. Disponível em: <http://www.wthreex.com/rup/>. Acesso em: 26 nov. 2012.

SILVEIRA, J. F. A. I**nstrumentação e segurança de barragens de terra e enrocamento.** [S.l.]: Oficina de Textos, 2006.

STONEBRAKER, M. **C-store: Acolumn-oriented dbms.** *VLDB*, páginas 553–564, Trondheim, Norway, 2005.

WREMBEL, R. **Data Warehouses and OLAP:Concepts, Architectures and Solutions** IRM Press, 1 edition, 2007.

APÊNDICES:

1 FASE DE INICIAÇÃO – WORKFLOW MODELO DE NEGÓCIO

1.1 VISÃO

Sistema de Monitoramento de Barragens DamMonit
Visão

Versão 1.0

Histórico da Revisão

Data	Versão	Descrição	Autor
14/10/12	1.0	Versão inicial do artefato	Adeildo Fernandes

Índice Analítico

1 Introdução

Neste documento serão definidas e analisadas necessidades e características que servirão de base para o desenvolvimento do projeto DamMonit.

Inicialmente será descrita a problemática que justifica a necessidade de desenvolvimento deste projeto, na sequência serão abordados os principais envolvidos e usuário, definindo seus papéis no desenvolvimento, utilização e manutenção do sistema a ser gerado, e por fim será dada uma definição em alto nível do que se espera que seja produzido ao final da elaboração do projeto.

2 Posicionamento

2.1 Descrição do Problema

Atualmente técnicos e engenheiros de segurança de barragens da hidrelétrica de 4 Gargantas não dispõem de uma ferramenta eficiente para a análise dos dados obtidos a partir da instrumentação instalada na barragem da usina. O atual sistema utilizado pelos envolvidos, em função da massiva quantidade de dados gerados, não atende de maneira satisfatória as necessidades dos envolvidos, além de ser centralizado, ou seja, não possui uma interface web que possa disponibilizar os dados através da internet.

Essas características impactam diretamente na qualidade das análises e estimativas de riscos da barragem, fazendo com que a qualidade das mesmas fiquem aquém do necessário para garantir a integridade física de toda a barragem.

Uma solução adequada para esse problema seria a criação de um sistema para o monitoramento de barragens que pudesse ser acessado via internet e que possuísse uma arquitetura de armazenamento de dados que pudesse atender os requisitos de performance exigidos pelos problemas os quais o sistema visa solucionar.

3 Descrições dos Envolvidos e Usuários

Nesta seção serão descritos os principais envolvidos na modelagem de negócios, manutenção e utilização do sistema.

3.1 Resumo dos Envolvidos

Nome	Descrição	Responsabilidades
Administradores	São os analistas de sistemas da hidrelétrica de 4 Gargantas.	Este envolvido irá garantir a manutenção do sistema como um todo. Ele ficará responsável pela criação de contas, recuperação de backups, monitoramento do ambiente computacional e demais situações pertinentes ao bom funcionamento do sistema.

Engenheiros	São os engenheiros responsáveis pela segurança de barragens da hidrelétrica.	Estes envolvidos participarão diretamente do processo de modelagem das regras de negócio do sistema a ser implementado.

3.2 Resumo dos Usuários

Nome	**Descrição**	**Responsabilidades**
Usuário do sistema	Todos os engenheiros e técnicos envolvidos com a segurança da barragem da hidrelétrica.	Esses usuários irão receber os dados coletados a partir da instrumentação instalada na barragem de Quatro Gargantas.

4 Visão Geral do Produto

Ao final do processo de desenvolvimento espera-se que como produto final um sistema para o monitoramento da instrumentação instalada na barragem da hidrelétrica de 4 Gargantas que possua uma performance de acesso aos dados satisfatória e que será disponibilizado em ambiente distribuído (web) cuja administração será centralizada (realizada pelo analistas da hidrelétrica).

1.2 CASOS DE USO NEGOCIAIS

Sistema de Monitoramento de Barragens DamMonit
Casos de Uso Negociais

Versão 1.0

Histórico da Revisão

Data	Versão	Descrição	Autor
14/10/12	1.0	Versão inicial do artefato	Adeildo Fernandes

Índice Analítico

1 Introdução

Neste documento será realizada uma descrição macro das funcionalidades do sistema. Estas funcionalidades foram dividias em três módulo a fim de simplificar a compreensão do sistema. Tais módulos serão descritos a seguir.

2 Módulo do Administrador

Este módulo irá contemplar todas as atividades relacionadas a administração do sistema. Este módulo irá contemplar a manutenção de usuários, instrumentos e locais da hidrelétrica. Compõem esse módulo os seguintes casos de uso:

UC002 – Manter Usuários → Cadastros e manutenção de usuários;
UC003 – Manter Locais → Cadastro e manutenção de locais;
UC004 – Manter Instrumentos → Cadastro e manutenção de instrumentos.

3 Módulo do Monitorador

Este módulo irá contemplar todas as atividades que são realizadas pelos monitoradores que farão uso do sistema. Este módulo irá contemplar o acesso e possibilidade de realizar monitoramento sobre os dados de leitura dos sensores da barragem da hidrelétrica. Compõem esse módulo os seguintes casos de uso:

UC001 – Realizar Login → Autentica usuário no sistema;
UC005 – Exportar Dados → Realiza a exportação dos dados do sistema para um formato específico passível de utilização por ferramentas de mineração de dados.
UC008 – Realizar Pesquisas de Leituras → Possibilita ao usuário ter contato com os dados de leitura dos instrumentos.

4 Módulo do Sistema

Este módulo será responsável por realizar tarefas relacionadas ao bom funcionamento do sistema desenvolvido. Este módulo irá contemplar ações que visão manter a consistência da base de dados do sistema. Compõem esse módulo os seguintes casos de uso:

UC006 – Sincronizar Dados → Realiza a consistência entre a base de dados orientada a linha e a orientada a coluna;
UC007 – Gerenciar Leitura dos Instrumentos → Alimenta a base com os dados fornecidos pelos instrumentos.

1.3 GLOSSÁRIO

Sistema de Monitoramento de Barragens DamMonit
Glossário

Versão 1.0

Histórico da Revisão

Data	Versão	Descrição	Autor
14/10/12	1.0	Versão inicial do artefato	Adeildo Fernandes

Índice Analítico

1 Introdução

Este documento é usado para definir a terminologia específica do domínio do problema, explicando termos que podem ser desconhecidos do leitor das descrições de caso de uso ou de outros documentos do projeto.

2 Definições

Row-Store

Modelo de armazenamento de dados tradicional, também conhecido como modelo relacional. Neste modelo todos os atributos são armazenados contiguamente em disco.

Column-Store

Modelo de armazenamento de dados não relacional o qual particiona verticalmente uma relação tradicional em sub-relações a fim de otimizar o processo de I/O entre memória principal e disco.

Piezômetro

Instrumento que mede a pressão hidrostática (também conhecida como poro pressão) do local onde foi instalado. Os piezômetros podem ser elétricos ou mecânicos.

Barragem

Podem ser denidas como construções humanas que criam obstáculos que impedem o fluxo natural das águas e tem como finalidades a conservação e acúmulo de água para a geração de energia elétrica, irrigação agrícola, abastecimento de água as cidades ou indústrias e prevenção de inundações, fazendo o controle do nível da água

Instrumentação de Barragens

A instrumentação representa um dos aspectos mais importantes quando se refere a Segurança de Barragens. É por meio dos dados fornecidos a partir das leituras realiza das nos instrumentos instalados nas barragens que é possível realizar análises a respeito das condições em que a mesma se encontra, além de poder fazer detecções precoces de problemas em potencial que a barragem pode apresentar ao longo do tempo. Leituras de instrumentos realizadas de forma correta e em épocas apropriadas são fundamentais para determinar o desempenho e a segurança de uma barragem

1.4 REGRAS DE NEGÓCIO

Sistema de Monitoramento de Barragens DamMonit
Regras de Negócio

Versão 1.0

Histórico da Revisão

Data	Versão	Descrição	Autor
14/10/12	1.0	Versão inicial do artefato	Adeildo Fernandes

Índice Analítico

1 Introdução

Neste documento serão Listadas todas as regras de Negócio que compõem o sistema DamMonit. O conjunto de regras de negócio do sistema será apresentado a seguir.

2 Regras de Negócio

R1. No momento que o aluno estiver preenchendo sua senha os caracteres da mesma dever ser visualizados como ‘*’ no campo “Senha”.

R2. Para inserir um instrumento é necessário informar obrigatoriamente o tipo do instrumento, a localização, e a data de instalação.

R3. Para inserir um usuário é necessário informar obrigatoriamente nome, login e senha.

R4. Para inserir um local é necessário informar obrigatoriamente a descrição, a latitude e longitude e a cota onde será instalado.

R5. Os arquivos .arf deverão ter o seguinte formato:
atributo1:valor1, atributo2:valor2, … , atributoN:valorN;
R6. O nome do arquivo a ser gerado deverá ter o seguinte padrão:
DamMonit_dia_mês_ano_hora_minuto.arf.

R7. A rotina de sincronização deve ser realizada periodicamente a cada 10 minutos;

R8. Em caso de falha o caso de uso deve esperar 20 segundos e ser executado novamente, até no máximo de 5 vezes.

R9. A rotina de gerenciamento de leitura de instrumentos deve ser realizada periodicamente a cada minuto;

R10. Em caso de falha o caso de uso deve esperar 20 segundos e ser executado novamente, indeterminadamente.

R11. Se não preencher o campo Local o sistema deve pesquisar por todos os locais cadastrados;

R12. Se não preencher o campo Instrumento o sistema deve pesquisar por todos os instrumentos cadastrados;

R13. O sistema deverá realizar os seguintes tipos de pesquisa: Todos os dados, Média, Moda, Mínimo, Máximo, Contagem e Soma;

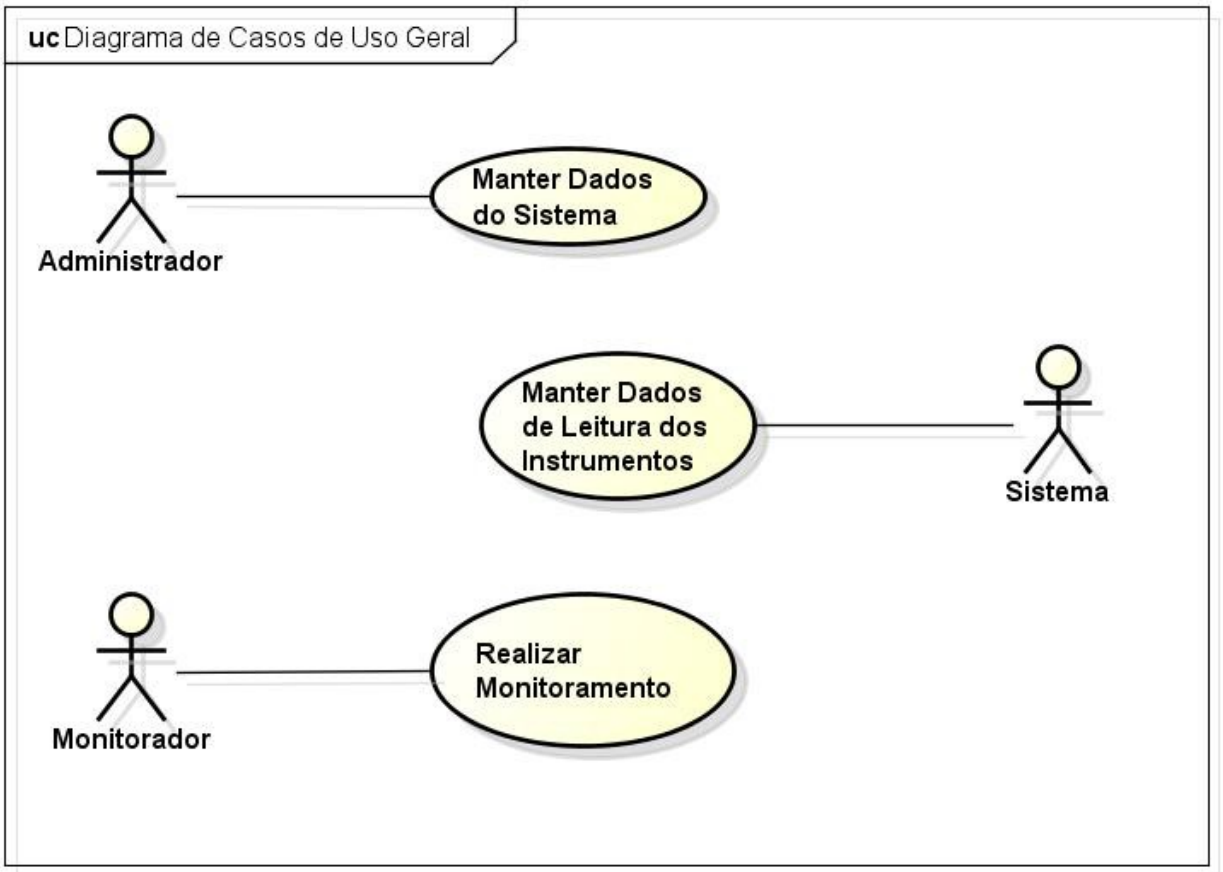
R14. Se a pesquisa for do tipo Todos os dados deve-se bloquear este campo;

R15. Se os campos de data não forem preenchidos o sistema trás os últimos 20 registros;

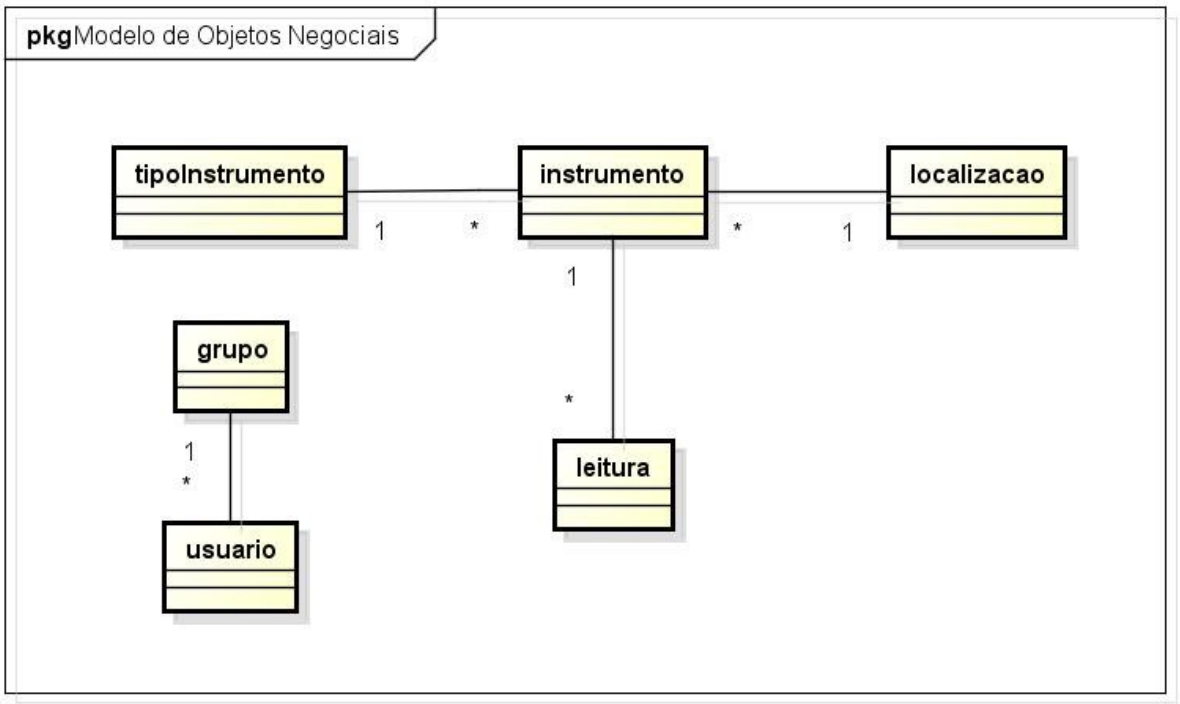
R16. Os campos de preenchimento obrigatório são: Tipo Instrumento, Tipo Pesquisa e Parâmetro de Pesquisa.

2 FASE DE ELABORAÇÃO – ITERAÇÃO 1 – WORKFLOW DE REQUISITOS

2.1 CASOS DE USO – DIAGRAMA DE CASOS DE USO PRINCIPAIS



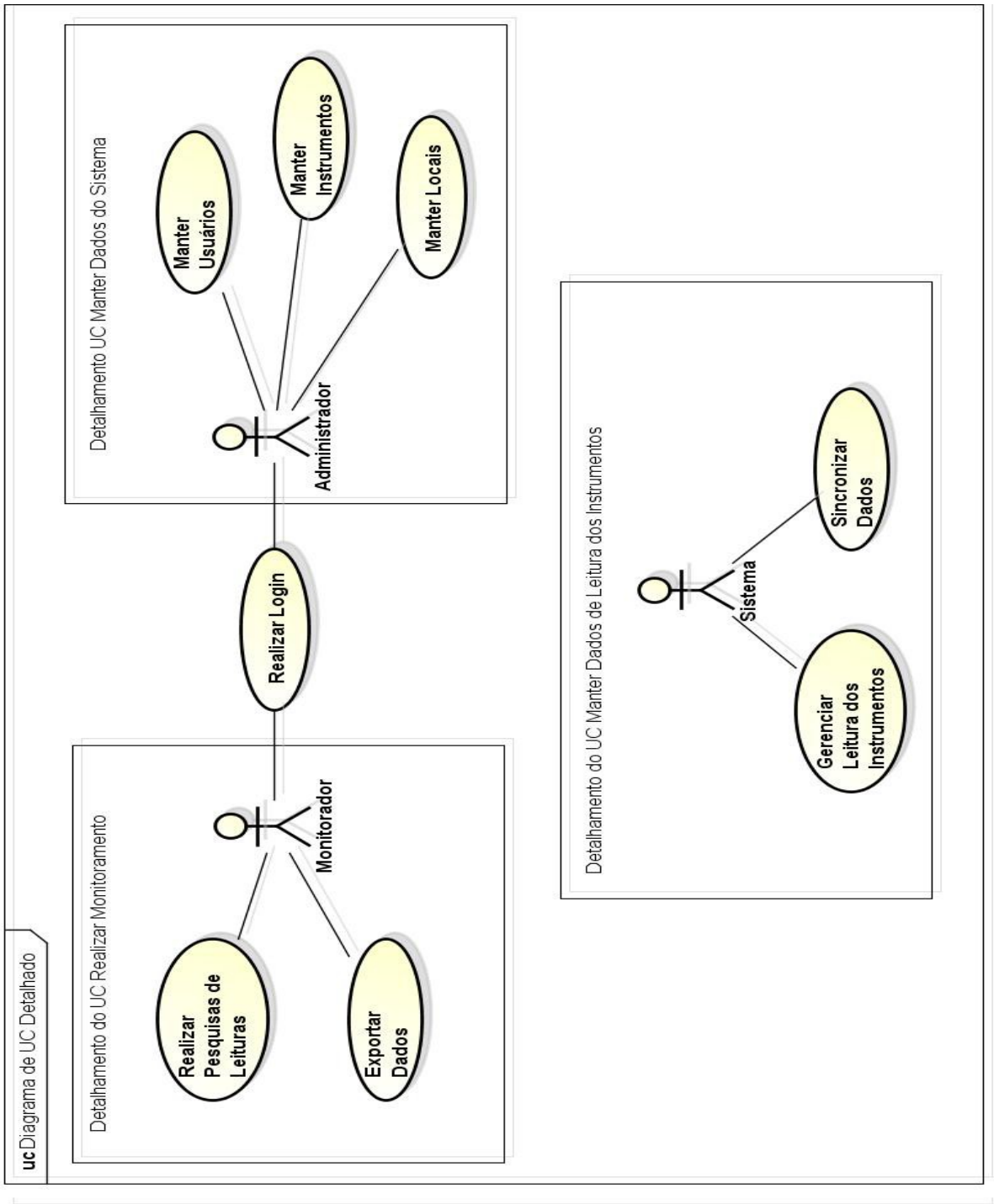
2.2 MODELO DE OBJETOS NEGOCIAIS



3 FASE DE ELABORAÇÃO – ITERAÇÃO 1 – WORKFLOW DE ANÁLISE E DESIGN

3.1 DIAGRAMA E ESPECIFICAÇÃO DETALHADA DOS CASOS DE USO

3.1.1 CASOS DE USO – DIAGRAMA DE CASOS DE USO DETALHADO



3.1.2 ESPECIFICAÇÃO DOS CASOS DE USO

DamMonit UC001 – Realizar Login

Controle do Documento

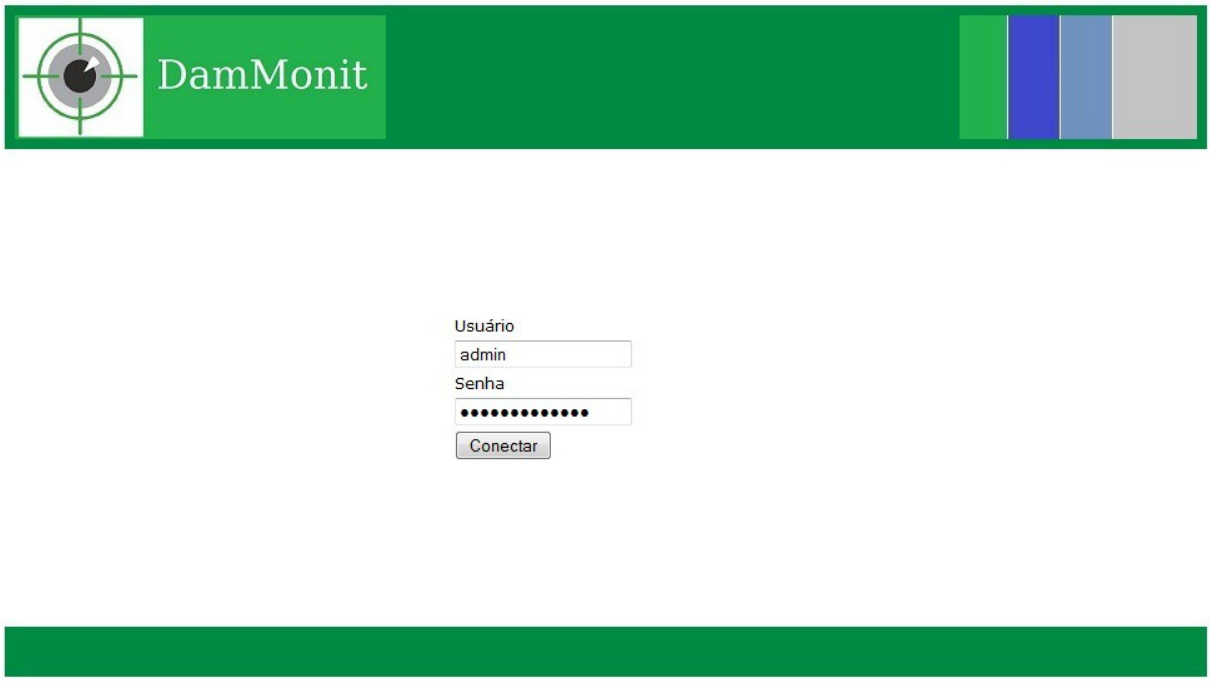
Versão	Autor	Data	Descrição
1.0	Adeildo	22/10/12	Elaboração

Descrição

Este caso de uso serve para autenticar o usuário no sistema DamMonit.

Data Views

DV1 – Tela Realizar Login.



DV2 – Tela de Entrada do Sistema DamMonit

Pré-condições
Não há.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:
1. Autenticar o usuário no sistema.

Ator Primário
Monitorador.

Fluxo de Eventos Principal
1. Sistema apresenta a tela (DV1);
2. O usuário preenche o campo usuário;
3. O usuário preenche o campo senha (R1);
4. O usuário pressiona o botão “Conectar”;
5. O sistema autentica o usuário (E1) (E2) ;
6. O sistema apresenta a tela de entrada do sistema DamMonit (DV2);
7. O caso de uso é finalizado.

Fluxos de Exceção
E1. Os campos “Usuário” e “Senha” (obrigatórios) não foram preenchidos:
1. O sistema retorna a mensagem “Para se autenticar no sistema é necessário informar o usuário e a senha”.
2. O sistema destaca os campos usuário e senha em vermelho e negrito.
3. O caso de uso é reiniciado.

E2. Falha na autenticação:
1. O sistema retorna a mensagem “Usuário ou Senha incorretos!”.

2. O sistema destaca os campos “Usuário” e “Senha” em vermelho.
3. O caso de uso é reiniciado.

Fluxos Alternativos
Não se aplica.

Regras de Negócio
R1. No momento que o aluno estiver preenchendo sua senha os caracteres da mesma dever ser visualizados como ‘*’ no campo “Senha”.

DamMonit UC002 – Manter Instrumentos

Controle do Documento

Versão	Autor	Data	Descrição
1.0	Adeildo	22/10/12	Elaboração

Descrição

Este caso de uso serve para cadastrar os instrumentos da barragem da hidrelétrica.

Data Views

DV3 – Tela de Listagem de Instrumentos.

DV4 – Telas de Cadastro de Instrumentos.

Pré-condições
Este caso de uso pode iniciar somente se:
1. O sistema tiver executado o caso de uso UC001 – Realizar Login.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:
1. O sistema deve armazenar o instrumento cadastrado.

Ator Primário
Administrador.

Fluxo de Eventos Principal
1. O sistema apresenta a lista de instrumentos já cadastrados no sistema (A1) (A2) (A3) (A4).
2. O caso de uso é finalizado.

Fluxos Alternativos
A1: Botão 'Novo' pressionado.
1. O usuário pressiona o botão 'Novo';
2. O sistema apresenta a tela de cadastro de instrumentos (DV4);
3. O usuário preenche os campos;
4. O usuário pressiona o botão Salvar;
5. O sistema persiste os dados do instrumento (A5) (E3);
6. O caso de uso é finalizado.

A2: Botão 'Editar' pressionado.
1. O usuário pressiona o botão de edição de instrumentos.
2. O sistema apresenta a tela (DV4) com os dados preenchidos;
3. O usuário altera os campos desejados;

4. O usuário pressiona o botão Salvar;
5. O sistema altera os dados do instrumento (A6) (E3);
6. O caso de uso é finalizado.

A3: Botão 'Exclusão' pressionado.
1. O usuário pressiona o botão de exclusão de instrumento.
2. O sistema apresenta a mensagem “Excluir o Instrumento?”;
3. O usuário escolhe a opção Sim (A7);
4. O sistema exclui o instrumento;
5. O caso de uso é finalizado.

A4: Botão 'Pesquisar' pressionado.
1. O usuário pressiona o botão de pesquisa de instrumentos.
2. O sistema filtra os instrumentos pelo filtro indicado;
3. O sistema apresenta os instrumentos encontrados;
4. O caso de uso é finalizado.

A5: Botão 'Voltar' pressionado.
1. O usuário pressiona o botão de cancelar inserção de instrumento.
2. O sistema retorna a listagem de instrumentos (DV3).
3. O caso de uso é finalizado.

A6: Botão 'Voltar' pressionado.
1. O usuário pressiona o botão de cancelar edição do instrumento.
2. O sistema retorna a listagem de instrumentos (DV3).
3. O caso de uso é finalizado.

A7: Botão 'Não' pressionado.
1. O usuário pressiona o botão de cancelar exclusão do instrumento.
2. O caso de uso é finalizado.

Fluxos de Exceção
E3. Campos Obrigatórios.
1. O sistema apresenta a mensagem “Os campos [nome do campo] são de preenchimento obrigatório” (R2);
3. O caso de uso é reiniciado.

Regras de Negócio

R2. Para inserir um instrumento é necessário informar obrigatoriamente o tipo do instrumento, o número de série, a localização, e a data de instalação.

DamMonit UC003 – Manter Usuários

Controle do Documento

Versão	Autor	Data	Descrição
1.0	Adeildo	22/10/12	Elaboração

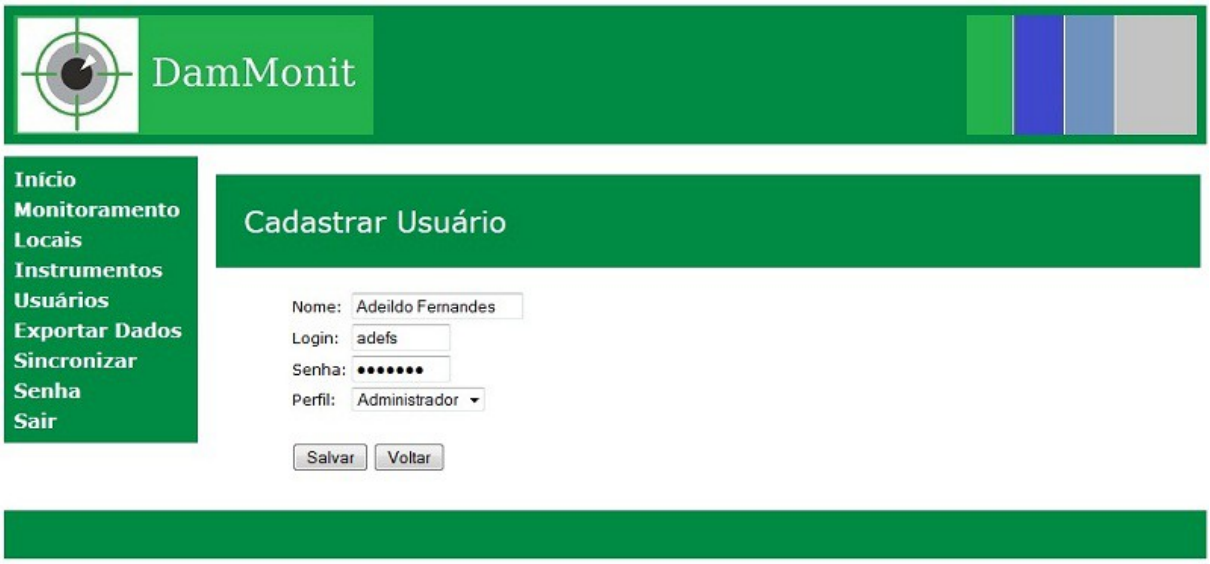
Descrição

Este caso de uso serve para manter os usuários do sistema DamMonit.

Data Views

DV5 – Tela de Listagem de usuários.

DV6 – Telas de Cadastro de usuários.



Pré-condições
Este caso de uso pode iniciar somente se:
1. O sistema tiver executado o caso de uso UC001 – Realizar Login.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:
1. O sistema deve armazenar o usuário cadastrado.

Ator Primário
Administrador.

Fluxo de Eventos Principal
1. O sistema apresenta a lista de usuários já cadastrados no sistema (A5) (A6) (A7) (A8).
2. O caso de uso é finalizado.

Fluxos Alternativos
A1: Botão 'Novo' pressionado.
1. O usuário pressiona o botão 'Novo';
2. O sistema apresenta a tela de cadastro de usuários (DV6);
3. O usuário preenche os campos;
4. O usuário pressiona o botão Salvar;
5. O sistema persiste os dados do usuário (A9) (E5);
6. O caso de uso é finalizado.

A2: Botão 'Editar' pressionado.
1. O usuário pressiona o botão de edição de usuários.
2. O sistema apresenta a tela (DV6) com os dados preenchidos;
3. O usuário altera os campos desejados;
4. O usuário pressiona o botão Salvar;

5. O sistema altera os dados do usuário (A10) (E5);
6. O caso de uso é finalizado.

A3: Botão 'Excluir' pressionado.

1. O usuário pressiona o botão de exclusão de usuário.
2. O sistema apresenta a mensagem “Excluir este usuário?”;
3. O usuário escolhe a opção Sim (A11);
4. O sistema exclui o usuário;
5. O caso de uso é finalizado.

A4: Botão 'Pesquisar' pressionado.

1. O usuário pressiona o botão de pesquisa de usuários.
2. O sistema filtra os usuários pelo filtro indicado;
3. O sistema apresenta os usuários encontrados;
4. O caso de uso é finalizado.

A9: Botão 'Voltar' pressionado.

1. O usuário pressiona o botão de cancelar inserção de usuário.
2. O sistema retorna a listagem de usuários (DV5).
3. O caso de uso é finalizado.

A10: Botão 'Não' pressionado.

1. O usuário cancelar edição do usuário.
2. O sistema retorna a listagem de usuários (DV5).
3. O caso de uso é finalizado.

A11: Botão 'Não' pressionado.

1. O usuário cancela a exclusão do usuário.
2. O caso de uso é finalizado.

Fluxos de Exceção

E5. Campos Obrigatórios.

1. O sistema apresenta a mensagem “Os campos [nome do campo] são de preenchimento obrigatório” (R3);
2. O caso de uso é reiniciado.

Regras de Negócio

R3.Para inserir um usuário é necessário informar obrigatoriamente nome, login e senha, e o perfil.

R3.1. Caso o usuário não preencha nenhum campo na busca deverão ser listados todos os usuários cadastrados.

DamMonit UC004 – Manter Locais

Controle do Documento

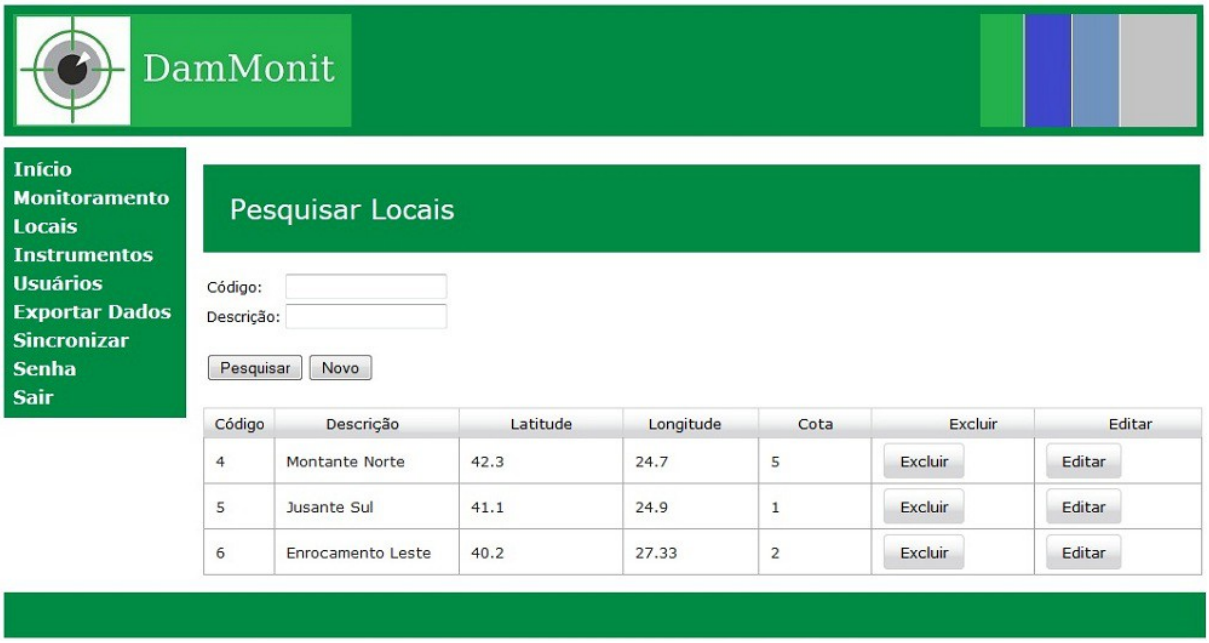
Versão	Autor	Data	Descrição
1.0	Adeildo	22/10/12	Elaboração

Descrição

Este caso de uso serve cadastrar os locais onde serão instalados os instrumentos da barragem.

Data Views

DV7 – Tela de Listagem de locais.



DV8 – Telas de Cadastro de locais.

Pré-condições
Este caso de uso pode iniciar somente se:
1. O sistema tiver executado o caso de uso UC001 – Realizar Login.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:
1. O sistema deve armazenar o local cadastrado e retorna a tela de listagem (DV7).

Ator Primário
Administrador.

Fluxo de Eventos Principal
1. O sistema apresenta a lista de locais já cadastrados no sistema (A12) (A13) (A14) (A15).
2. O caso de uso é finalizado.

Fluxos Alternativos
A12: Botão 'Novo' pressionado.
1. O usuário pressiona o botão 'Novo';
2. O sistema apresenta a tela de cadastro de locais (DV);
3. O usuário preenche os campos;
4. O usuário pressiona o botão Salvar;
5. O sistema persiste os dados do local (A16) (E7);
6. O caso de uso é finalizado.

A13: Botão 'Editar' pressionado.
1. O usuário pressiona o botão de edição de locais.
2. O sistema apresenta a tela (DV8) com os dados preenchidos;
3. O usuário altera os campos desejados;
4. O usuário pressiona o botão Salvar;

5. O sistema altera os dados do local (A17) (E7);
6. O caso de uso é finalizado.

A14: Botão 'Exclusão' pressionado.
1. O usuário pressiona o botão de exclusão de local.
2. O sistema apresenta a mensagem “Tem certeza que deseja excluir o local 'Nome do local'”;
3. O usuário escolhe a opção Ok (A18);
4. O sistema exclui o local;
5. O caso de uso é finalizado.

A15: Botão 'Pesquisar' pressionado.
1. O usuário pressiona o botão de pesquisa de locais.
2. O sistema filtra os locais pelo filtro indicado;
3. O sistema apresenta os locais encontrados (E8);
4. O caso de uso é finalizado.

A16: Botão 'Cancelar' pressionado.
1. O usuário pressiona o botão de cancelar inserção de local.
2. O sistema retorna a listagem de locais (DV7).
3. O caso de uso é finalizado.

A17: Botão 'Cancelar' pressionado.
1. O usuário pressiona o botão de cancelar edição do local.
2. O sistema retorna a listagem de locais (DV7).
3. O caso de uso é finalizado.

A18: Botão 'Cancelar' pressionado.
1. O usuário pressiona o botão de cancelar exclusão do local.
2. O caso de uso é finalizado.

Fluxos de Exceção
E7. Campos Obrigatórios.
1. O sistema apresenta a mensagem “Existem campos que deverão ser preenchidos. Por favor, preencha os campos que estão em destaque” (R1);
2. O sistema destaca os campos obrigatórios que não foram preenchidos em negrito e vermelho.
3. O caso de uso é reiniciado.

E8. Campo Consulta em Branco.
1. O sistema apresenta a mensagem “Para realizar a consulta de um local é necessário informar um parâmetro para o filtro”;
2. O sistema destaca o campo filtro em negrito e vermelho.
3. O caso de uso é reiniciado.
4.

Regras de Negócio

R4. Para inserir um local é necessário informar obrigatoriamente a descrição, a latitude e longitude e a cota onde será instalado.

DamMonit UC005 – Exportar Dados

Controle do Documento

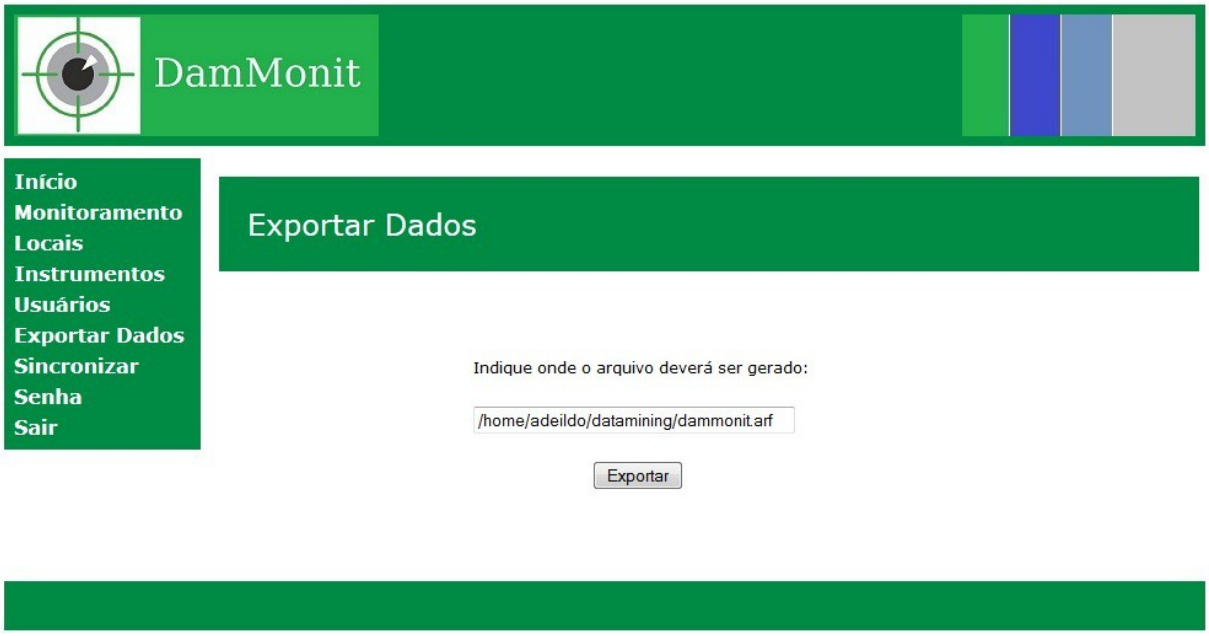
Versão	Autor	Data	Descrição
1.0	Adeildo	22/10/12	Elaboração

Descrição

Este caso de uso serve para exportar os dados de leitura dos instrumentos para um arquivo no formato .arf para que o mesmo seja utilizado em processos de mineração de dados.

Data Views

DV9 – Tela de Exportação de Dados



Pré-condições
Não se aplica.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:

1. Gerar um arquivo .arf no diretório escolhido pelo usuário.

Ator Primário
Monitorador

Fluxo de Eventos Principal
1.O sistema apresenta tela para escolha do local para salvar o arquivo a ser gerado;
2.O usuário determina local onde salvar o arquivo;
3.O usuário clica em 'OK' (E9);
4.Sistema gera o arquivo .arf (R5) (R6);
5.O caso de uso é finalizado.

Fluxos Alternativos
Não se aplica

Fluxos de Exceção
E9: Nenhum diretório é selecionado para salvar o arquivo.
1.O sistema apresenta a mensagem “É necessário escolher um diretório para armazenar o arquivo.”;
2.O caso de uso é reiniciado.

Regras de Negócio

R5. Os arquivos .arf deverão ter o seguinte formato:
atributo1:valor1, atributo2:valor2, … , atributoN:valorN;

R6. O nome do arquivo a ser gerado deverá ter o seguinte padrão:
DamMonit_dia_mês_ano_hora_minuto.arf.

DamMonit UC006 – Sincronizar Dados

Controle do Documento

Versão	Autor	Data	Descrição
1.0	Adeildo	22/10/12	Elaboração

Descrição

Este caso de uso irá realizar o sincronismo de dados entre a tabela 'leituras' e suas equivalentes no esquema *column-store*.
Data Views
Não se aplica.

Pré-condições
Este caso de uso pode iniciar somente se:
1.Haverem dados a serem sincronizados.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:
1.Deixar as tabelas armazenadas em *column-store* com os mesmos dados armazenados na tabela 'leituras'.

Ator Primário
Sistema

Fluxo de Eventos Principal
1.O sistema acessa a tabela que armazena os dados de leitura dos instrumentos (tabela 'leituras') (R7) (E10);
2.O sistema seleciona os registros que ainda não foram armazenadas no esquema *column-store* (E10).
3.O sistema percorre os registros e realiza a cópia dos mesmos para as tabelas equivalentes do esquema *column-store*. (E10)
4. Os registros da tabela 'leituras' que foram gravados com sucesso no esquema *column-store* tem o campo 'sincronizado' setado como '*true*'.

Fluxos Alternativos
Não se aplica.

Fluxos de Exceção
E10: Comunicação com o banco de dados falhou.
1.O sistema grava em log data data e hora da falha (R2);
2.O caso de uso é reiniciado**.**

Regras de Negócio

R7. A rotina de sincronização deve ser realizada periodicamente a cada 10 minutos;

R8. Em caso de falha o caso de uso deve esperar 20 segundos e ser executado novamente, até no máximo de 5 vezes.

DamMonit UC007 – Gerenciar Leitura dos Instrumentos

Controle do Documento

Versão	Autor	Data	Descrição
1.0	Adeildo	22/10/12	Elaboração

Descrição

Este caso de uso serve armazenar no sistema os dados gerados a partir das leituras dos instrumentos instalados na barragem.

Data Views
Não se Aplica

Pré-condições
Este caso de uso pode iniciar somente se:
1. O arquivo C:\DamMonit\dadosLeituras.txt existir.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:
1.Armazena os dados retirados do arquivo 'dadosLeituras.txt' na base de dados;
2.Apaga do arquivo 'dadosLeitura.txt' os dados que foram armazenados na base.

Ator Primário
Sistema

Fluxo de Eventos Principal
1.Sistema abre arquivo C:\DamMonit\dadosLeituras.txt (E11) (R9) (R10);
2.Sistema realiza leitura dos dados do arquivo;
3.Sistema armazena os dados lidos na base de dados (E12) (R10);
4.Sistema retira os dados armazenados na base do arquivo;
5.O caso de uso é finalizado.

Fluxos Alternativos
Não se aplica;

Fluxos de Exceção
E11: Arquivo não está disponível;
1. O sistema registra em log a data e hora da indisponibilidade do arquivo;
2. O caso de uso é reiniciado.

E12: Comunicação com o banco de dados falhou.
1.O sistema grava em log data e hora da falha de acesso a base de dados (R10);
2.O caso de uso é reiniciado**.**

Regras de Negócio

R9. A rotina de gerenciamento de leitura de instrumentos deve ser realizada periodicamente a cada minuto;
R10. Em caso de falha o caso de uso deve esperar 20 segundos e ser executado novamente, indeterminadamente.

DamMonit UC005 – Realizar Pesquisas de Leituras

Controle do Documento

Versão	Autor	Data	Descrição
1.0	Adeildo	22/10/12	Elaboração

Descrição

Este caso de uso tem a finalidade de realizar o monitoramento dos dados coletados pelos sensores da instrumentação da barragem hidrelétrica.

Data Views

DV10 – Tela de Monitoramento

Pré-condições
Este caso de uso pode iniciar somente se:
1. O sistema tiver executado o caso de uso UC001 – Realizar Login.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:

1. Exibir os detalhes da consulta realizada pelo monitorador.

Ator Primário
Monitorador

Fluxo de Eventos Principal
1. Sistema apresenta a tela (DV10);
2. O usuário preenche o campo Tipo Instrumento;
3. O usuário preenche o campo Local (R11);
4. O usuário preenche o campo Instrumento (R12);
5. O usuário preenche o campo Tipo de Pesquisa (R13);
6. O usuário preenche o campo Parâmetro de Pesquisa (R14);
7. O usuário preenche os campos Data Inicial e Data Final (R15);
8. O usuário pressiona o botão “Pesquisar” (E13);
5. O sistema realiza a pesquisa dos dados no banco de dados ;
6. O sistema apresenta o resultado da pesquisa na tela (DV10);
7. O caso de uso é finalizado.

Fluxos Alternativos
Não se aplica.

Fluxos de Exceção
E13. Campos Obrigatórios.
1. O sistema apresenta a mensagem “Existem campos que deverão ser preenchidos. Por favor, preencha os campos que estão em destaque” (R16);
2. O sistema destaca os campos obrigatórios que não foram preenchidos em negrito e vermelho.
3. O caso de uso é reiniciado.

Regras de Negócio

R11. Se não preencher o campo Local o sistema deve pesquisar por todos os locais cadastrados;

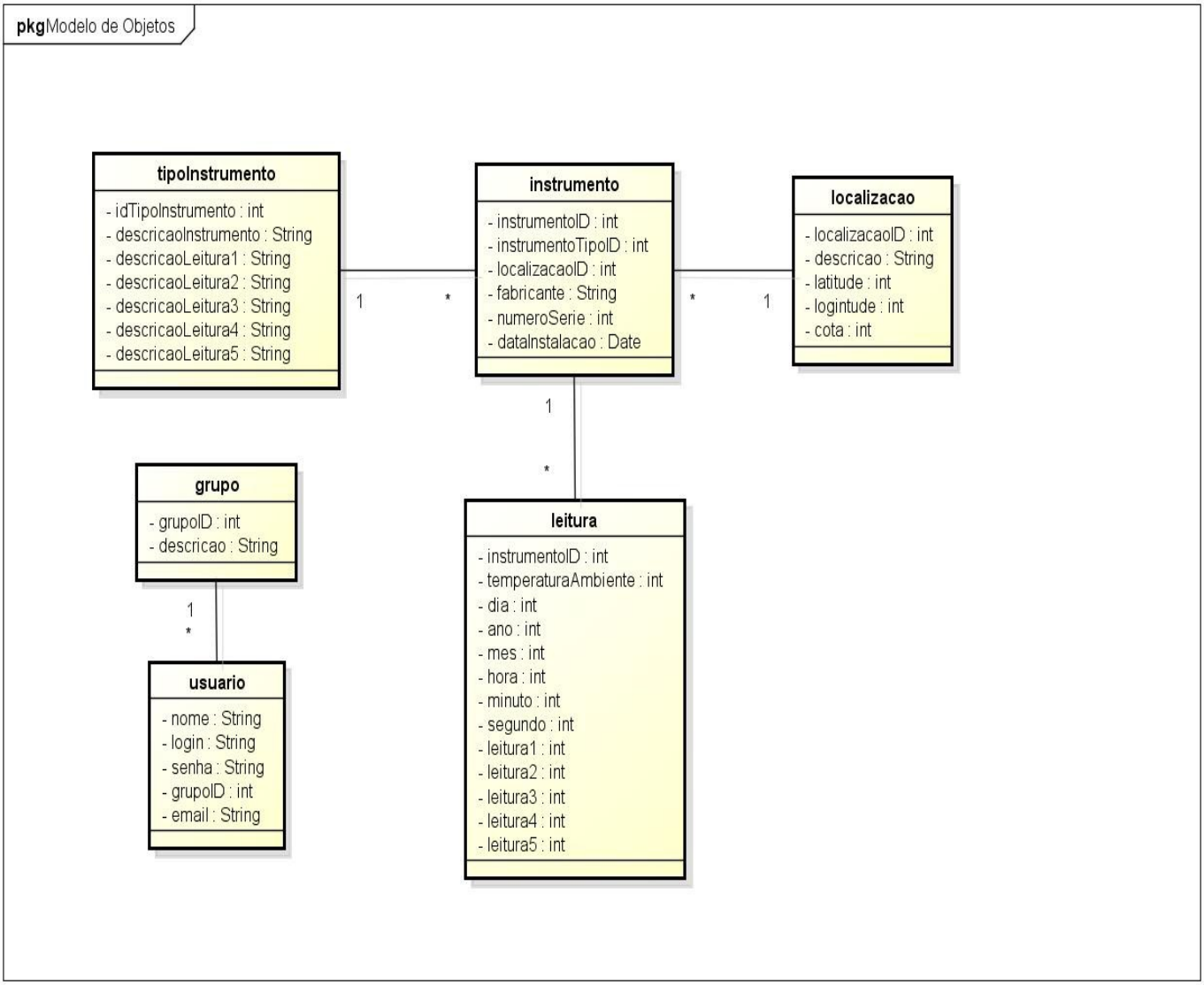
R12. Se não preencher o campo Instrumento o sistema deve pesquisar por todos os instrumentos cadastrados;

R13. O sistema deverá realizar os seguintes tipos de pesquisa: Todos os dados, Média, Moda, Mínimo, Máximo, Contagem e Soma;

R14. Se a pesquisa for do tipo Todos os dados deve-se bloquear este campo;

R15. Se os campos de data não forem preenchidos o sistema trás os últimos 20 registros;
R16. Os campos de preenchimento obrigatório são: Tipo Instrumento, Tipo Pesquisa e Parâmetro de Pesquisa.

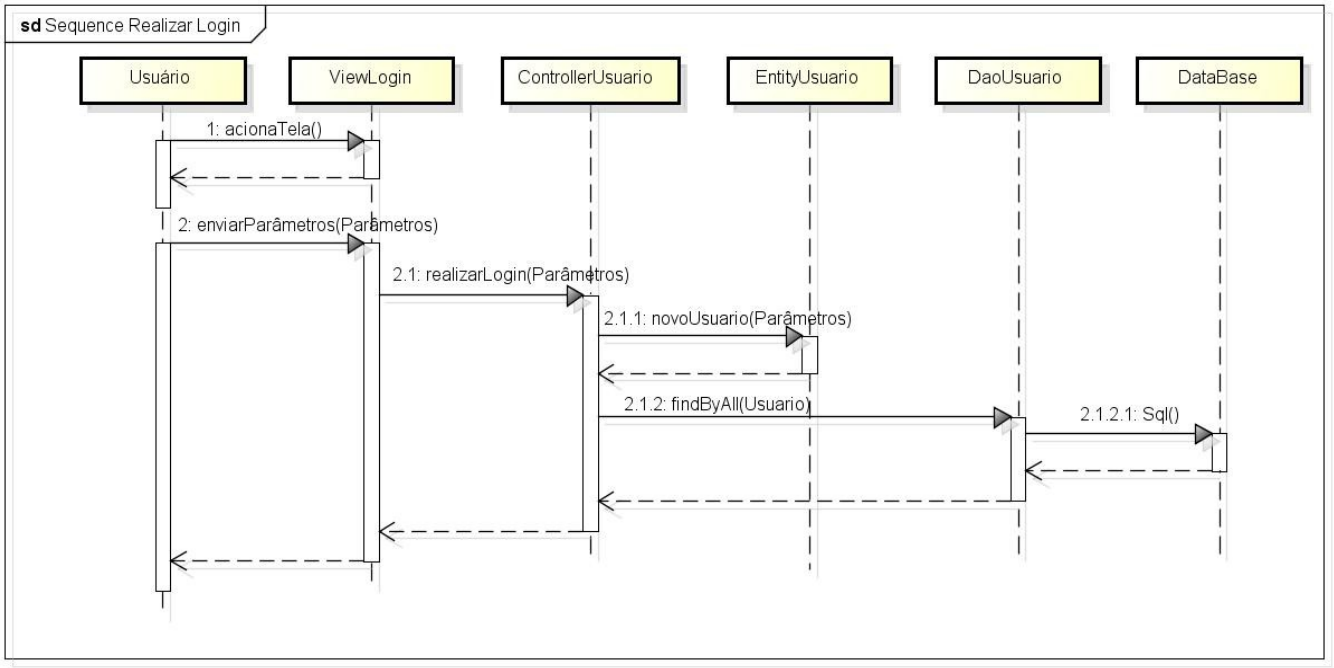
3.2 MODELOS DE OBJETOS



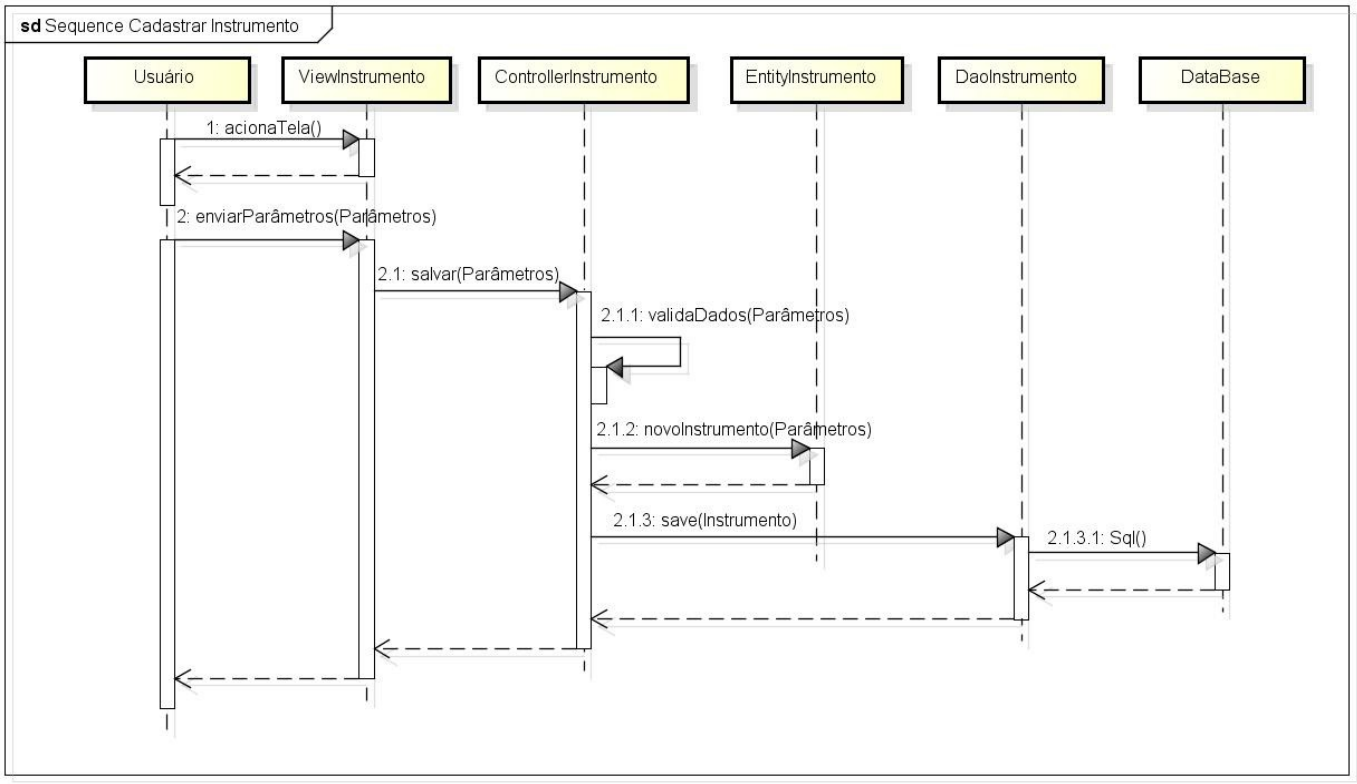
4 FASE DE ELABORAÇÃO – ITERAÇÃO 2 – WORKFLOW DE ANÁLISE E DESIGN

4.1 DIAGRAMAS DE SEQUÊNCIA

4.1.1 DIAGRAMA DE SEQUÊNCIA REALIZAR LOGIN

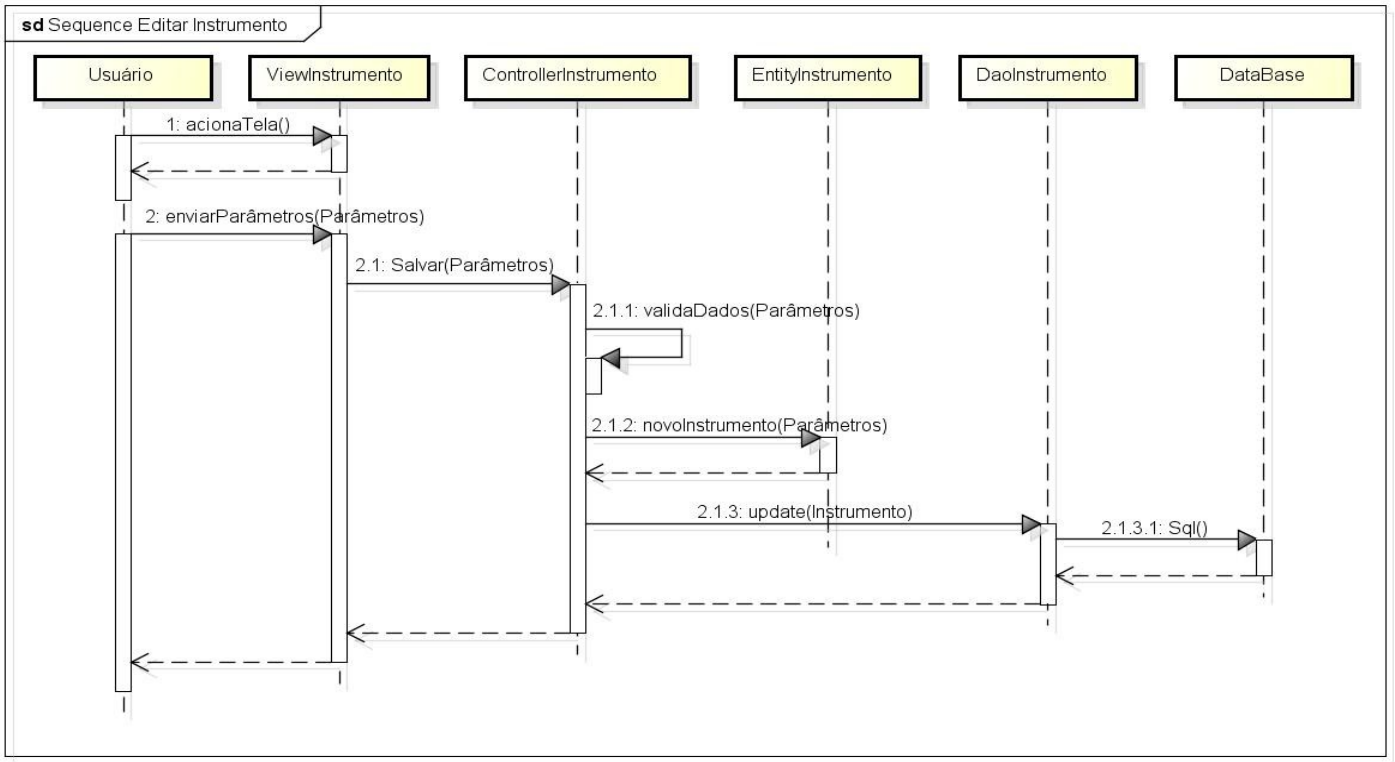


4.1.2 DIAGRAMA DE SEQUÊNCIA CADASTRAR INSTRUMENTO

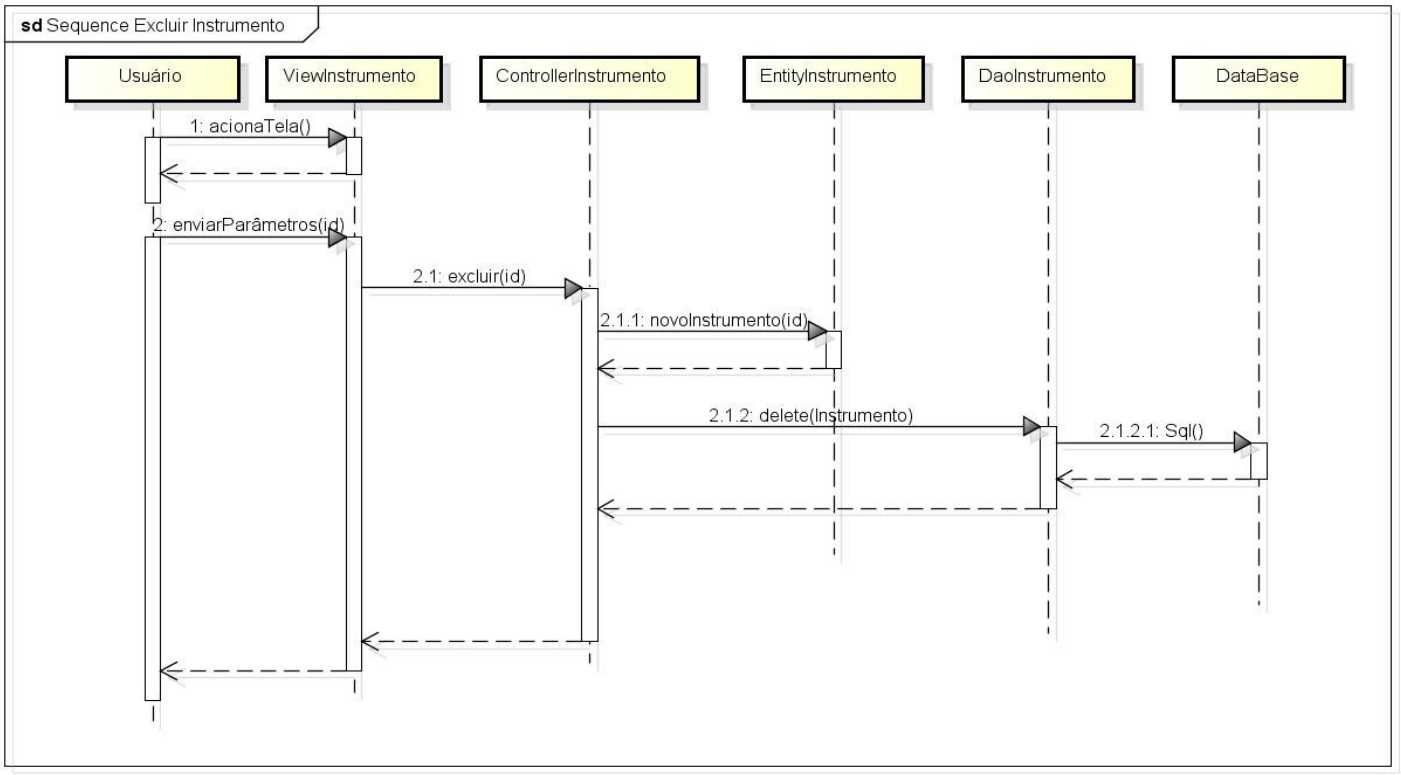


4.1.3

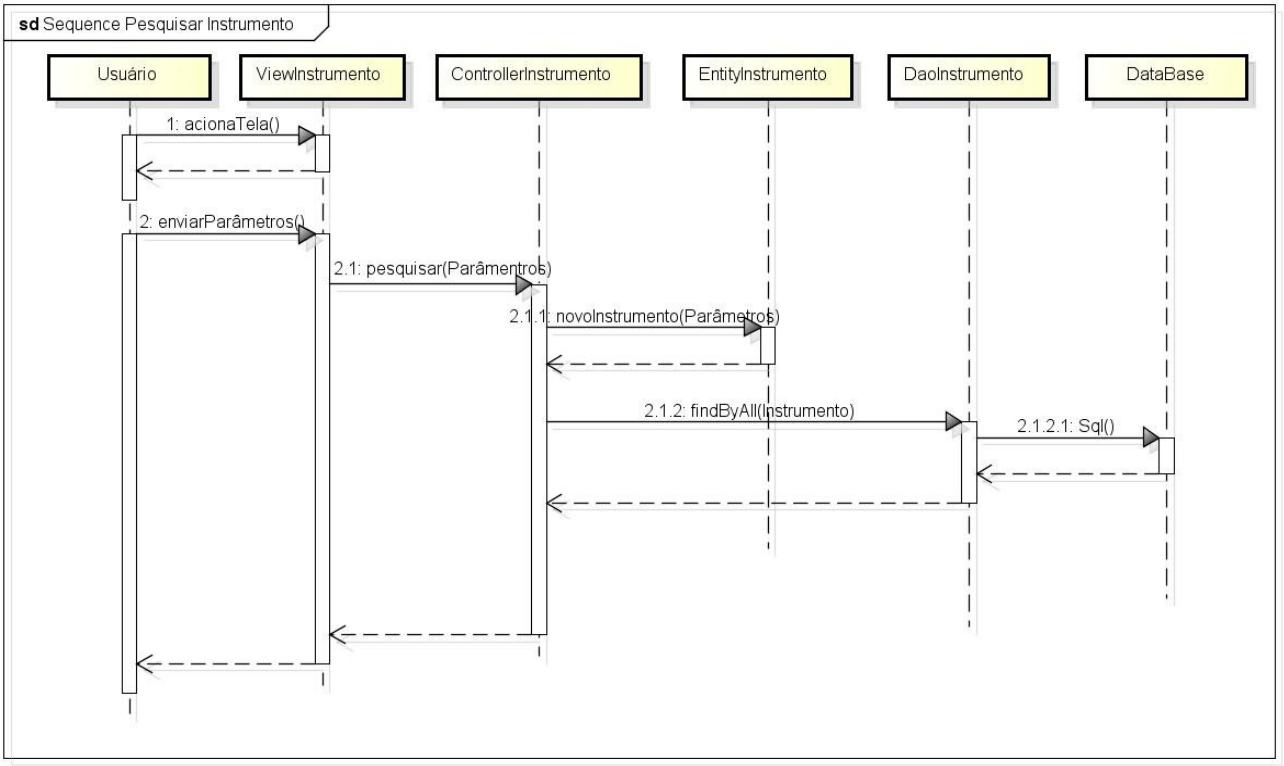
4.1.3 DIAGRAMA DE SEQUÊNCIA EDITAR INSTRUMENTO



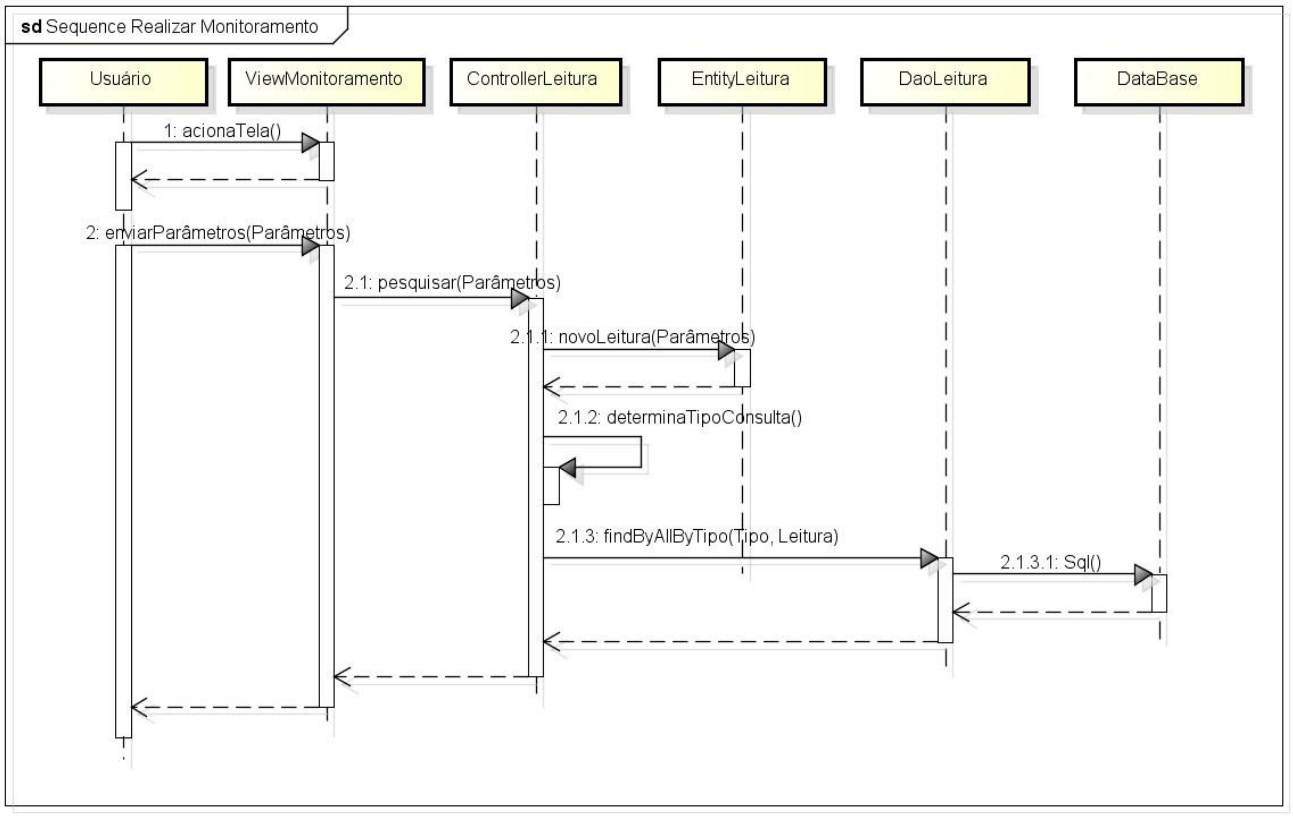
4.1.4 DIAGRAMA DE SEQUÊNCIA EXCLUIR INSTRUMENTO



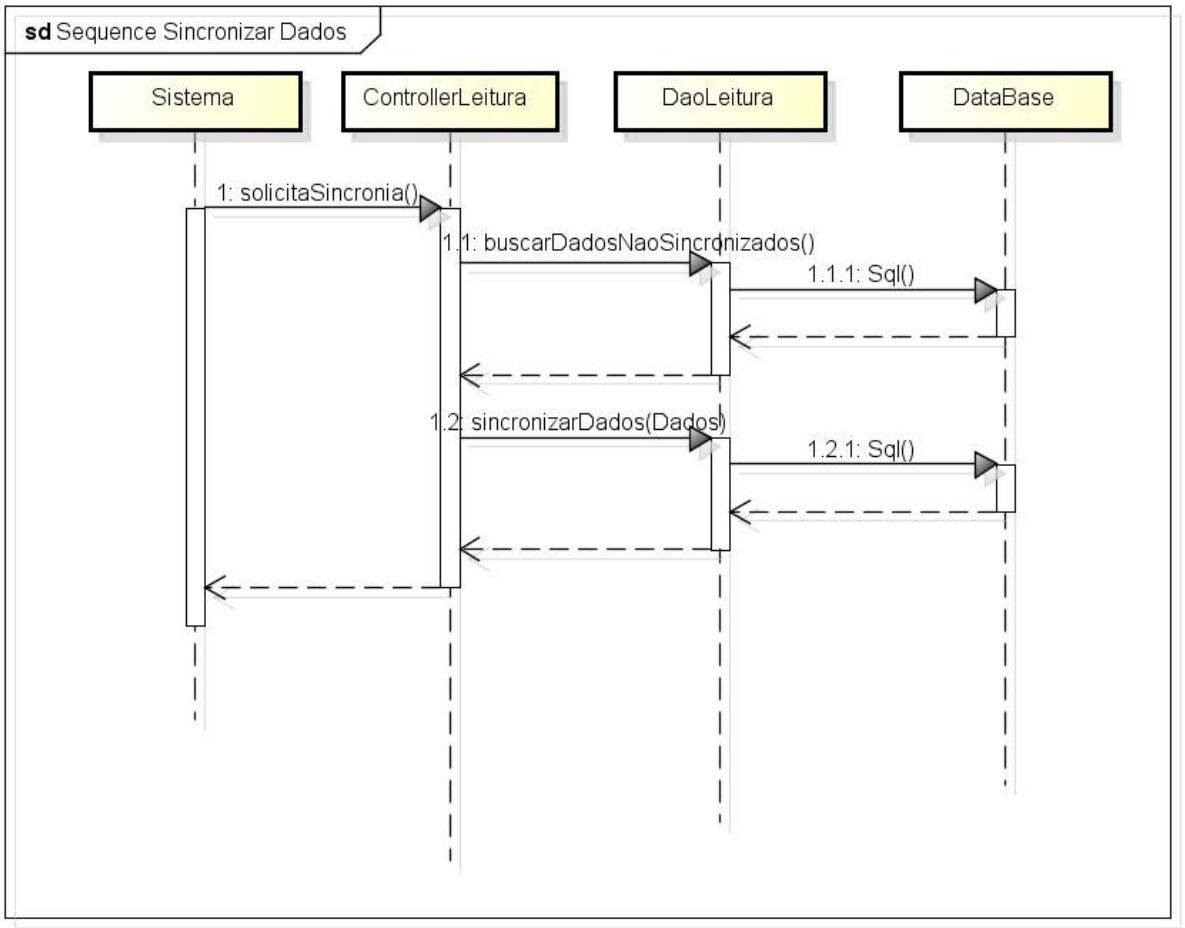
4.1.5 DIAGRAMA DE SEQUÊNCIA PESQUISAR INSTRUMENTO



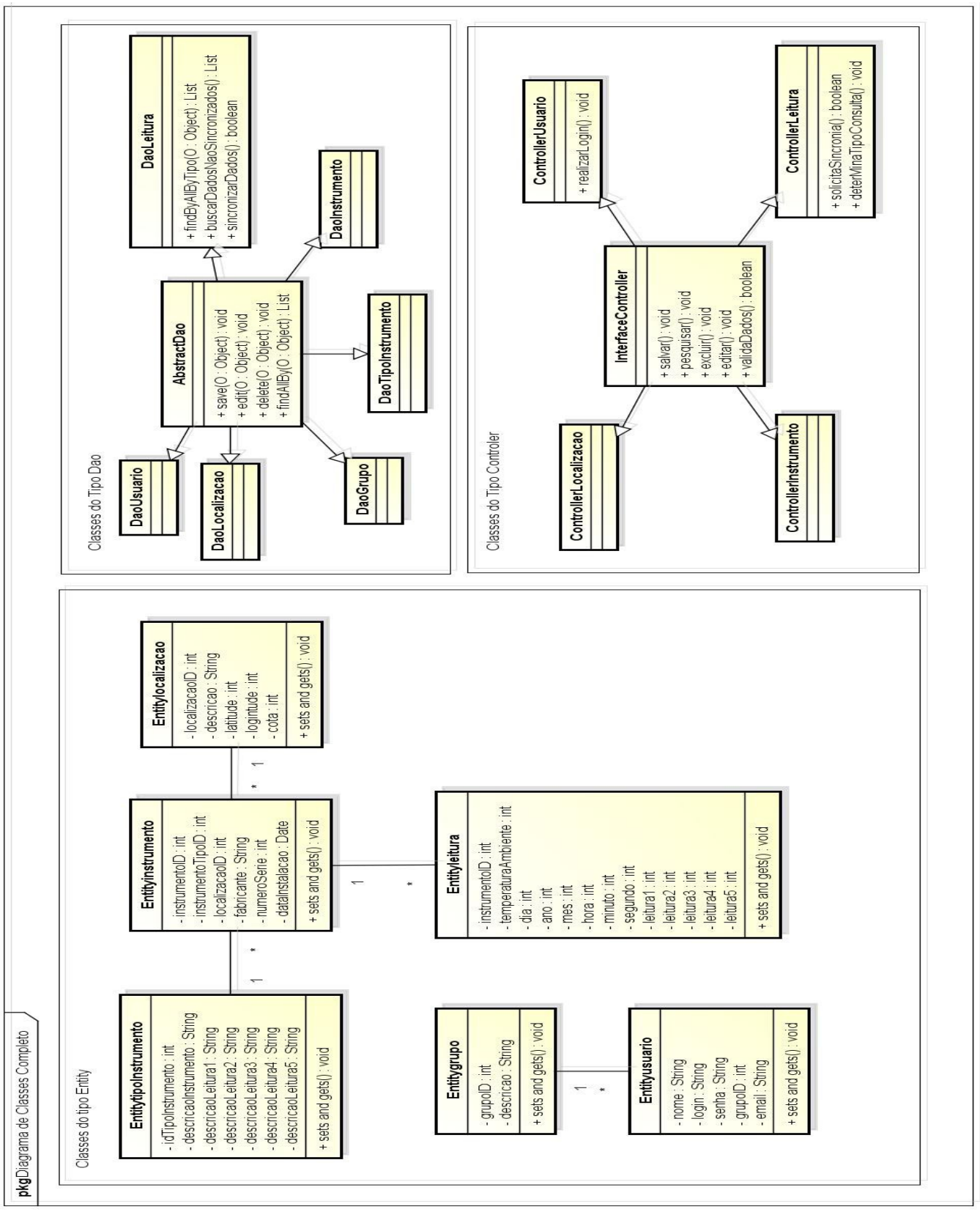
4.1.6 DIAGRAMA DE SEQUÊNCIA REALIZAR MONITORAMENTO



4.1.7 DIAGRAMA DE SEQUÊNCIA SINCRONIZAR DADOS

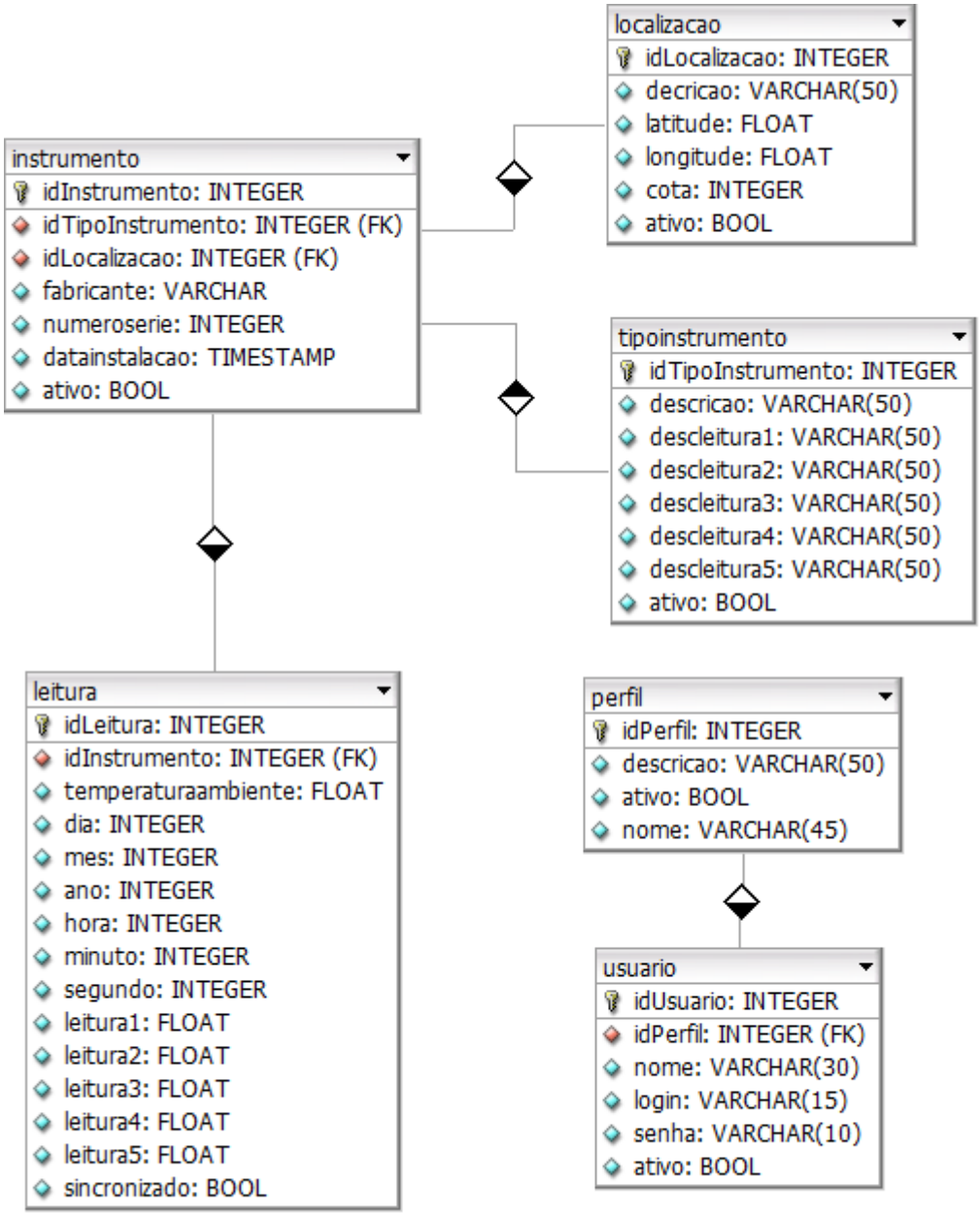


4.2 MODELO DE OBJETOS



4.3 MODELO FÍSICO DE DADOS

4.3.1 MODELO FÍSICO DE DADOS BASE RELACIONAL



4.3.4 MODELO FÍSICO DE DADOS BASE ORIENTADA A COLUNA